MULTIPLE CRITERIA OPTIMIZATION : AN EVALUATION OF TWO EVOLUTIONARY META-HEURISTIC METHODS

by

T. DIWAKAR SRINIVAS

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JANUARY, 1997

MULTIPLE CRITERIA OPTIMIZATION : AN EVALUATION OF TWO EVOLUTIONARY META-HEURISTIC METHODS

A Thesis Submitted

in Partial Fulfilment of the Requirements

for the Degree of

Master of Technology

by

T. DIWAKAR SRINIVAS

to the

DEPARTMENT OF INDUSTRIAL AND MANAGEMENT ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JANUARY, 1997

IME-1997-M-SRI-MUL

Entered in System

An
20-4-98

CERTIFICATE

It is to certify that the work contained in the thesis entitled "MULTIPLE CRITERIA EVOLUTIONARY META-OPTIMIZATION: AN EVALUATION OF TWO HEURISTIC METHODS" by Mr. T. DIWAKAR SRINIVAS has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

[illegible]

Prof. T. P. Bagchi
Industrial Management and Engineering Department
Indian Institute of Technology
Kanpur - 208016

7th January 1998

ACKNOWLEDGMENTS

First and foremost, I express my sincere gratitude to Professor Tapan P. Bagchi, who helped me profusely at various stages of this thesis work. I am grateful for his timely suggestions and encouragement which enabled me to complete this work.

I am also thankful to Prof. Kalyanmoy Deb for clarifying my doubts whenever I approached him despite his busy schedule.

This acknowledgment would not be complete without the mention of Mr. Neeraj Kumar, Ph.D. (IME) for all the help he gave me unsparingly — be it the discussions on GA and all its cousins, or the long hours spent in rendering this report so good looking.

I would like to thank all the professors, my colleagues and the whole IME family who made my stay in IIT a memorable one.

I am also deeply indebted to my parents, brother and sister for their keen interest evinced in my work. Their constant support and deep concern have helped me to cross many hurdles.

T. DIWAKAR SRINIVAS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Management decisions today typically involve not only many decision variables, but also a multiplicity of *objectives*. Furthermore, these objectives are often in conflict, leading to complexity in decision making. Traditional multi-objective methods usually do not optimize all objectives simultaneously. Frequently, the multiple objectives are converted into a single objective using weights *a priori* and then the problem is solved as a single-objective optimization problem. Deb and Srinivas (1995) proposed a meta-heuristic (called NSGA) for multi-objective optimization which produces Pareto optimal solutions. This method uses a nondominated sorting technique to produce the solutions. However, by design NSGA does not deliberately preserve good solutions from generation to generation. In the present work a new *elitist* algorithm (ENGA) is developed and then evaluated using typical numerical multiobjective optimization problems. The key difference between ENGA and NSGA is that ENGA performs an additional nondominated sorting of the parents+progeny pool and subsequently uses the top 50% of the population to form the next generation. As done in recent GA applications, the statistical design-of-experiments approach is used to find the optimum GA parameter settings for implementing ENGA. Many individual factor effects and their interactions are found to be significant. The present work then empirically compares the performances of NSGA and ENGA using two different optimization problems as test beds and by conducting the Wilcoxon's signed-rank test. In both the cases ENGA shows superior convergence. The thesis concludes with suggestions for future work.

Chapter 1

MULTIOBJECTIVE OPTIMIZATION

A characteristic of many formal techniques that support quantitative decision making is the selection of the *best alternative* with respect to certain figures of merit. The decision maker's challenge multiplies when he/she confronts situations involving multiple decision goals. In this chapter the problem of optimizing multiple objectives is recalled. The common solution methods for solving the multiobjective optimization problem and the concepts of Pareto optimality and "efficient solutions" are reviewed. The chapter concludes with the citation of the nondominated sorting genetic algorithm (NSGA) [12], a recently proposed meta-heuristic solution procedure that has been evaluated and extended in the present work.

1.1 Multiple Criteria Decision Making

In recent years the nature of decision problems has changed considerably. Serious doubts have been raised as to the adequacy of many classical models of decision situations and their solution techniques. For instance, numerous factors now affect the typical modern industrial enterprise, leading not only to many decision variables but also to the multiplicity of *objectives* in decision making. Adding to this complication is the fact that such objectives are not only multiple but also often conflicting.

In a manufacturing organization, for example, apart from its overall objectives of profit and market share maximization, each department may have specific objectives to attain (Table 1.1).

Table 1.1: Multiple Decision Objectives in a Manufacturing Enterprise

Department	**Objective(s)**
Budget	Cost minimization
Production	Production output maximization Production time minimization Resource utilization
Quality Control	Product quality maximization Rework minimization
Personnel	Minimization of hiring and firing
Marketing	Uninterrupted supply of products to customers

Many applications of modelling such Multiple Criteria Decision Making (MCDM) problems seem to adverently and explicitly express organizational aspirations in terms of only a single criterion. It has been noted that this may indeed be a misinterpretation of reality. In recent years, therefore, the MCDM methodology has come into prominence. Its tools and techniques have been developed and verified and subsequently applied to a variety of problems which in great proportion control the success of industrial and service enterprises.

The general multi-objective problem requiring the optimization of *k* objectives simultaneously may be formulated as follows.

$$\text{Max (min)} \quad Z_j = f_j(X), \; j = 1,2,\ldots,k$$

Subject to

$$g_i(X) \le b_i, \; i = 1,2,\ldots,m$$

$$X \ge 0$$

where X is a vector of decision variables and $g_i(X)$ are the inequality constraints. In multicriterion optimization, because the objectives are conflicting, there does not exist a single unique solution which is globally maximum or globally minimum with respect to all the conflicting objectives. The increase in any one of these objectives will decrease the others and vice versa.

1.2 A Necessary Condition: Conflicting Criteria

A necessary condition of MCDM is the presence of *more than one* criterion. The sufficient condition is that the criteria must be *conflicting* in nature. Thus the MCDM problem can be defined as the problem with at least two conflicting criteria and there are at least two alternative solutions [51].

Criteria are said to be in conflict if the full satisfaction of one will result in impairing the full satisfaction of other(s). The criteria are said to be "strictly" conflicting if increase in satisfaction of one results in a decrease in satisfaction of the other. The sufficient condition of MCDM, however, does not necessarily stipulate "strictly" conflicting criteria.

The techniques of MCDM are developed based on the fact that all the objectives are dispensable and that all can be traded off although some may be more important than the rest.

1.3 Classification of Multiobjective Problems

There are several ways to classify the different approaches to multiobjective optimization. Adulbhan and Tabucanon [1] classified the techniques into three main approaches based on the way the initial multiobjective problem is transformed into a mathematically manageable format. These approaches are, respectively, (a) conversion of secondary objectives into constraints, (b) development of a single combined objective function, and (c) treatment of all objectives as constraints.

Hwang, Masud, Paidy and Yoon [27], on the other hand, proposed a different classification. They based their grouping of techniques according to the *stage* at which information from the decision maker is needed by the analyst. The classification is divided into four approaches: (a) no articulation of preference information, (b) "a priori" articulation of preference information, (c) progressive articulation of preference information, and (d) "a posteriori" articulation of preference information.

Among others, who have published survey papers on multiobjective optimization, are Johnson [29], Roy [42], Cochrane and Zeleny [9], Lietmann and Marzollo [34] and Hwang and Masud [28].

1.4 Solution Methods

1.4.1 Single Objective Approach

A simple way of handling multiobjective optimization problems with k objectives would be to optimize one objective (the most "important" one) and to treat the resulting k-1 "secondary" objectives into constraints [51]. This may be done by specifying a maximum (for minimization) or minimum (for maximization) level of attainment for each of the secondary objectives. Thus the multiobjective problem is converted into and subsequently solved as a single objective optimization problem, as follows.

Maximize $\quad Z_1 = f_1(X)$

Subject to

$$\begin{aligned} g_i(X) &\leq b_i, \; i = 1, 2, \ldots, m \quad \text{(original constraints)} \\ f_j(X) &\geq z_j^1, \; j = 2, 3, \ldots, k \quad \text{(additional constraints)} \\ X &\geq 0 \end{aligned}$$

where z_j^1 are specified minimum levels of attainment allowed of the remaining objectives. In this all objectives are assumed of maximizing nature. Although this method is practicable, there are certain cases where the approach gives no defined feasible region after the introduction of additional (k-1) constraints.

1.4.2 Global Criterion Method

This method [51] develops a global objective function which is made up of the ratio of sum of the deviations of the values of the individual objective functions from their respective ideal values to that of the ideal values. Thus, from the original k objective functions, a single function is formulated and the problem tantamounts to solving a single objective optimization. This method does not require explicit information on the relative importance of the objectives. Additionally, there is less subjectivity involved in the formulation process.

$$\text{Minimize} \quad F = \sum_{l=1}^{k} \left[\frac{f_l(x^*) - f_l(x)}{f_l(x^*)} \right]^p$$

Subject to

$$g_i(X) \leq 0, \quad i = 1,2,....,m$$

$$X \geq 0$$

where $f_l(x^*)$ is the value of objective function l at its individual optimum x^*, $f_l(x)$ is the function itself, p is an integer valued exponent that serves to reflect the importance of the objectives, and g_i is the function of constraint i.

1.4.3 Utility Function Method

The utility function method [51] converts the multiobjective optimization problem into a single objective problem in the following form:

$$\text{Maximize} \qquad Z = F[f_1(X), f_2(X),..., f_k(X)]$$

Subject to

$$g_i(X) \leq 0, \quad i = 1, 2,..., m$$

$$X \geq 0$$

where F is the utility function of the multiple objectives, representing the decision maker's preferences. If F is properly determined, the solutions obtained will be expected to ensure the decision maker's maximum satisfaction. However, at times the determination of F can be extremely difficult. Further, F may appear in many forms, the most common forms assuming the decision maker's utility function to be additively separable with respect to the objectives. The additive utility function method therefore converts the objective functions into one with the following form:

$$\text{Maximize } Z = \sum_{j=1}^{k} w_j f_j(X)$$

Here w_j indicates the relative importance the decision maker attaches to objective j and it must be specified "a priori".

1.4.4 Minimum Deviation Method

This method [51] is applicable when the analyst has information about the optimal values of the objectives but their relative importance is not known. The method aims at finding the best compromise solution which minimizes the sum of individual objectives' *fractional* deviations obtained from individual optimum values. The fractional deviation of an objective refers to the ratio of the deviation of a value of that objective from its individual optimal solution and its *maximum* deviation. The maximum deviation of an objective is obtained from the difference between its individual optimal solution and its least desirable solution, which would correspond to the individual optimal solution of one of the other objectives.

1.4.5 Goal Programming Approach

Goal Programming (GP) is a widely used multicriteria method that requires both ordinal and cardinal information for multiple objective decision making [51]. In GP, *deviation variables* (from goals) with assigned priorities and weights are minimized instead of optimizing the objective criterion directly as done, for instance, in LP. The general form of goal programming may be expressed as follows:

$$\text{Minimize } Z = \sum_{i=1}^{m} (p_{oi} d_i^+ + p_{ui} d_i^-)$$

Subject to

$$\sum_{j=1}^{n} a_{ij} x_j + d_i^- - d_i^+ = b_i, \quad i = 1, 2, \ldots, m$$

$$x_j,\ d_i^-,\ d_i^+ \geq 0, \quad i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, n$$

where x_i are the basic variables in the goal equations, b_i are the targets or goals, a_{ij} are coefficients of the basic variables, d_i^- represent the underachievement of goal i, d_i^+ represent the overachievement of goal i, P_{ui} is the priority associated with d_i^-, and P_{oi} is the priority associated with d_i^+. If overachievement is acceptable, d_i^+ can be eliminated from the objective function Z and if underachievemnt is acceptable, d_i^- can be eliminated. If goal i must be achieved exactly as defined, then both d_i^- and d_i^+ must both appear in the objective function. If goals are classified in R ranks, priority factors P_r ($r = 1,\ldots.,m$) should be assigned to the deviation variables. The priority factors have the following relationships: $P_r >> N\ P_{r+1}$, which implies that the multiplication of N, however large it may be, cannot make P_{r+1} greater than or equal to P_r. The algorithm is executed by a modified simplex method [51].

From the above it becomes evident that conventional methods essentially convert a multiple objective optimization problem into a single objective problem by some means or the other. No method takes care of optimizing all the objectives simultaneously to obtain a *set* of solutions (such as the "Pareto optimal" or "efficient" solutions discussed later in this chapter) which offer a way to simultaneously "satisfice" all objectives [45]. Recently, heuristic methods which engage the Pareto-optimality concept to produce multiple nondominated solutions to multiobjective problems have been proposed. One such heuristic, which is tested and extended in the present work, is Nondominated Sorting Genetic Algorithm (NSGA), created by Deb and Srinivas [12] NSGA is a metaheuristic search approach based on concept of niche formation in natural biological evolution. NSGA is actually an extension of Simple Genetic Algorithm (SGA) created by Holland [24], a method for heuristically optimizing a single objective. SGA has rapidly gained fellowship among both researchers and practitioners in the past decade.

1.5 Multiple Criteria Optimization Redefined

A recently proposed method for treating the analytical phase of the multiple criteria decision process is called multiple criteria optimization or, in short, multiobjective optimization [43]. According to this viewpoint, multiple criteria optimization contains two key concepts: (1) Pareto optimality and (2) the preferred decision (preferred solution). In general, the decisions with Pareto optimality are not uniquely determined, unlike, for instance, what goal programming produces. In multiobjective optimization problems, there usually exist many solutions that are optimal in the Pareto sense. From this viewpoint, due to such plurality of optimal decisions, the most desirable decision may be selected *after* one has generated the Pareto optimal or nondominated solutions. The final solution thus selected as the most desirable, or at least the best compromised solution, is called the *preferred solution.*

Multi-objective optimization thus redefined is not purely a maximization or minimization problem. It is a mixture of several conflicting maximization or minimization problems which boils down to that of "satisficing" these conflicting objectives [45].

Briefly, the solution to the multi-objective problem (Section 1.1) is obtained a *set* of solutions (called the Pareto optimal or "efficient" solutions borrowing the notion of "efficiency" of production functions in economics), the selection of particular solution from this set depends on decision maker's preferences for the different objectives. Thus the real challenge for the analyst in the multi-objective optimization would now be to provide the decision maker a set of such "better" (or "efficient") solutions (Figure 1.1), each of which are superior to one another in some aspect, so that the decision maker may then apply some a posteriori criteria to select one solution among these that best suits his requirements.

1.6 The Concept of Pareto Optimality and "Efficient" Solutions

In optimization, the "optimal" solution is one which attains maximum values (or minimum values) of all of several objectives simultaneously. Thus the solution x^* is optimal if and only if $x^* \in s$ and $f_l(x^*) \geq f_l(x)$ for all l and for all $x \in s$ where s is the feasible region.

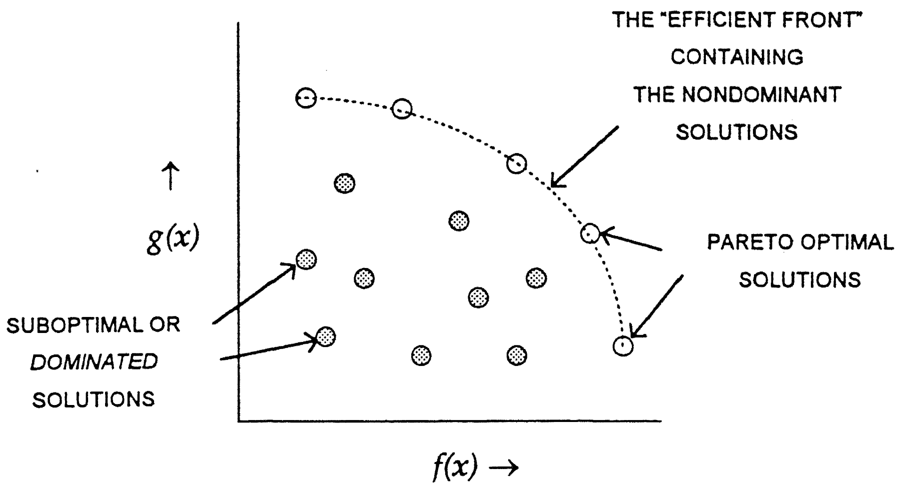
THE "EFFICIENT FRONT" CONTAINING THE NONDOMINANT SOLUTIONS
PARETO OPTIMAL SOLUTIONS
SUBOPTIMAL OR *DOMINATED* SOLUTIONS
↑ g(x)
f(x) →

Figure 1.1: The Efficient Front in a Bi-objective Maximization Problem

However, in case of multi-objective optimization with conflicting objectives, there is *no* unique optimal solution. A simple optimal solution may exist here only when the objectives are non-conflicting. Thus for conflicting objectives one may at best obtain what economists call "efficient" solutions (Figure 1.1). An *efficient* solution (also called a Pareto optimal solution) is one in which no increase can be obtained in any of the objectives without causing a simultaneous *decrease* in at least one of the remaining objectives [31]. Thus solution x^* is efficient to the multi-objective problem if and only if there does not exist any $x \in s$ such that $f_l(x) \geq f_l(x^*)$ for all l and $f_l(x) > f_l(x^*)$ for at least one l. The formal representation here may be given as follows.

Consider the general multiple objective optimization problem consisting of a number of equality and inequality constraints such as

Min (Max) $F_i(X)$ $i = 1, 2..., n$

Subject to

$$G_j(X) \leq 0, \ j = 1, 2, ..., J$$

$$H_k(X) \leq 0, \quad k = 1, 2, \ldots, K$$

here **X** is a n-dimensional vector having n decision variables and the objectives $\{F_i(X)\}$ are in onflict with each other (If $\{F_i(X)\}$ are not in conflict, a single solution, when feasible, will be ptimum). Solutions to optimizing $\{F_i(X)\}$ are expressed as *non-dominated* solutions. hapter 2 of this thesis introduces metaheuristic search methods known as genetic algorithms n the optimization literature. Chapter 3 describes the nondominated sorting technique ncorporated in genetic algorithms and applies it to a difficult multiobjective engineering lesign problem. Chapter 4 introduces an enhanced version of NSGA, dubbed ENGA and ddresses the selection of optimal settings of parameters to achieve its faster convergence. In hapter 5 the performances of the two algorithms NSGA and ENGA are statistically ompared and the results are presented. Chapter 6 describes the computer code mplementation of ENGA. For minimization problems, solution vector **X**(1) composed of decision variables $x_1, x_2, \ldots, x_n$ "dominates" (i.e. is superior to) **X**(2) (**X**(1) < **X**(2)) iff no component of **X**(2) is smaller in magnitude than **X**(1) and at least one component of **X**(2) is greater than **X**(1). In case of maximization problems the situation is reverse. A point (x_1, $x_2, \ldots, x_n$) in the decision space is dominated if any {X} component x_i is not greater than that of a corresponding non-dominated point. The non-dominated solutions are called the Pareto optimal solutions in economics and decision science. These constitute the efficient solutions to the multiple objective problem.

6.1 The ε-Constraint Method to Produce Efficient Solutions

general, the multiple criteria optimization problem involving *well-behaved* (continuous and fferentiable) objective functions and constraints [43, page 38] may be stated in the following rm

$$\underset{x \in X}{\text{Max}} \quad [f_1(x), f_2(x), \ldots, f_m(x)]$$

Subject to

$$g_j(x) \quad \leq \quad 0, \quad j = 1, 2, \ldots, J$$

he corresponding ε-constraint problem **P$_k$(ε)** to deliver Pareto optimal solutions here is rmulated as follows

$$\underset{x \in X}{\text{Max}} \quad f_k(x)$$

Subject to

$$f_i(x) \quad \geq \quad \varepsilon_i^1, i = 1, 2, \ldots, m, \; i \neq k$$

$$g_j(x) \quad \leq \quad 0, \quad j = 1, 2, \ldots, J$$

vhere parameter $\varepsilon_{i,r}^1$ is calculated as $f_{i,r}^1 = f_{i,r-1}^1$ - $\varepsilon_{i,r}$, $\varepsilon_{i,r} > 0$. 'i' denotes the number of imes of repetitive calculation (r = 1,......,T). $f_{i,0}$ is an initial value of the i-th objective unction $f_i(x)$, which has been optimized for $x \in X$.

n this method, the maximization problem stated above having m objective functions is reduced to the scalar-valued maximization problem having only the k-th objective function $f_k(x)$, in which all the (m-1) objective functions are treated as constraints. The problem $P_k(\varepsilon)$ is repeatedly solved for $x_r \in X$ corresponding to the value of the constraint constant $\varepsilon_i{}^1{}_r$ which is parametrically varied by shifting with the parameter $\varepsilon_{i,r} > 0$ in each iteration. The set of solutions obtained in each iteration derives the noninferior (Pareto optimal) solution set.

We note that rather few objectives and constraints in problems in management science are *well-behaved*, a precondition for the ε-constraint method to apply.

Recently interest has grown in techniques for producing Pareto optimal solutions. For instance, when the objective functions and constraints are well-behaved, it is possible to find a Pareto optimal solution set using ε-constraint method [43]. Many real problems, however, do not possess this "well-behaved" quality. As we describe in this work, heuristic methods have been now designed to find the non-dominated solutions in such situations.

Chapter 2

GENETIC ALGORITHMS

Genetic Algorithms (GAs) (algorithmic search procedures inspired by natural evolution of organisms) have received a great deal of attention regarding their potential as optimization techniques for complex functions [16]. This chapter provides an introduction to GAs. The appropriate evaluation (fitness) function and the types of (genetic) operators are described. The chapter concludes with a summary of relative merits reported of GAs when compared to the traditional techniques used for optimization and the more advanced variations proposed for GAs.

2.1 What are Genetic Algorithms?

GAs are heuristics that use randomized as well as *directed smart search* to seek the global optima — inspired by evolutionary processes through which life is believed to have evolved to its well-adapted forms [22]. Two such other good heuristics now available are simulated annealing [10] and tabu search [21]. GAs, however, have been noted by researchers to be more general and context-independent.

Genetic Algorithms (GAs) are a set of global search and optimization methods that is fast gaining popularity in solving engineering optimization problems. These belong to the class of metaheuristic methods that evolve superior solutions to global optimization problems *adaptively*. GAs systematically evolve a population of candidate solutions-with the objective of reaching the "best" solution-by using process inspired by natural genetic variation and natural selection as explained by Holland [24] and formalized by Goldberg [22] and others.

Natural selection eliminates one of the greatest hurdles in software design for optimization: specifying in advance all the features of a problem and the actions a program should take to deal with them [25]. Holland argued and demonstrated that genetic algorithms make it possible to explore a far greater range of potential solutions to a problem than do conventional programs.

Typically a GA can be characterized by the following components [47]

- A genetic representation (or an encoding) for the feasible solutions to the optimization problem
- A population of encoded solutions
- A fitness function that evaluates the optimality of each solution
- Genetic operators that generate a new population from the existing population.
- Control parameters.

The complete solution string s (of length l) is represented as (s_{l-1} s_{l-2} ... s_2 s_1 s_0). The decoded value of the substring s_i is calculated here as

$$\Sigma\, 2^i s_i \qquad \text{where } s_i \in \{0,1\}$$

2.2.1 Fitness Function

Unlike simulated annealing and tabu search, which explore the solution space sequentially, GAs work with *populations* of solutions and attempt to guide the search toward improvement [18]. GAs emulate or mimic the "survival of the fittest" principle of nature to accomplish the search process. To achieve maximization, a *fitness function* F(x) is first derived from the problem's objective function f(**x**) and one sets F(**x**) = f(**x**). For minimization, several different transformations are possible. One popular transformation is F(**x**) = 1/(1+f(**x**)), without altering the location of the minimum, but converting the original minimization problem into a maximization problem. Depending on the problem, the measure of fitness could be business profitability, game payoff, error rate, or any number of other criteria. High-quality strings mate; low-quality ones perish. As generations pass, strings associated with improved solutions will predominate.

2.3 Search Process is Population-Based

GAs begin with a population of randomly selected initial solutions — a population of random strings representing the problem's decision variables. Thereafter, each of these strings is evaluated to find its fitness. If some specified criterion is met, the search is stopped. If not, the initial population is subjected to "genetic evolution" — to procreate the next generation of candidate solutions. Figure 2.1 shows the algorithm of Simple Genetic Algorithm (SGA).

The genetic process of procreation uses the initial population as input. The members of the population are "processed" by three main GA *operators* — reproduction, crossover and mutation- to create the progenies (the next generation of candidate solutions to the optimization problem at hand).

```
begin
        t ← 0;
        initialize population P(t);
        evaluate members of P(t);
        while (termination criteria not met) do
            begin
            t ← t + 1;
            reproduction (to form the mating pool);
            crossover;
            mutation;
            evaluate members of P(t + 1);
            select (to form the progenies);
            end
end;
```

Figure 2.1: Algorithm Simple GA

2.3.1 The Reproduction Operator

This operator first selects good parent solutions or strings to form the mating pool. The essential idea is to select strings of above average fitness from the existing population and insert their multiple copies in the mating pool, in a probabilistic manner. One popular reproduction operator uses fitness proportionate selection in which a parent solution is selected to move to the mating pool with a probability that is proportional to its own fitness. Another selection operator gaining popularity for its effectiveness in selecting above average strings is the *Tournament Selection operator*. In this two strings are selected at random and their fitness values are compared. The one with better fitness gets copied into the mating pool. This is called the *binary selection*. For selection any number of strings can be compared simultaneously.

2.3.2 The Crossover Operator

The power of GAs arises from the crossover operator[47]. In the crossover operation new strings are created by exchanging information among the strings present in the mating pool. Here two strings are picked from the mating pool randomly and some portions of these strings are exchanged between them. There exists now a large variety of crossover operators in GA. The essential idea behind each of these operators is to combine the good features of the parents and to incorporate these features in the child chromosome. The following paragraphs give an overview of various crossover operators.

2.3.2.1 Single Point Crossover

The commonly used version of crossover is the single point crossover. In single point crossover a "cross site" is chosen along the string length and all the bits to the right side of this crossing site are exchanged between the two parent strings. Since the exact location of the crossing site is not fixed a priori, it is selected at random.

2.3.2.2 Two-point Crossover

This is the simple extension of single point crossover. Here instead of one site, two cross sites are chosen at random along the string length and the information (i. e. , bits) trapped between these sites are exchanged between the parents.

2.3.2.3 Uniform Crossover

With this operator a random site is not chosen. Instead, every bit is considered by turn for possible exchange between parents. For each bit a coin is flipped with some probability and if the flip turns out to be head, only then that corresponding bit is exchanged between the two parents. Thus the extent of information exchange obtained with this operator is much larger than that obtained by single or two point crossover. Uniform crossover compared both theoretically and empirically with single point and two-point crossovers has been shown to be superior in many cases [49].

2.3.2.4 Simulated Crossover

It treats the population of a genetic algorithm as a conditional variable to a probability density function that predicts the likelihood of generating samples in the problem space [50]. Gilbert[50] considers one specific version of simulated crossover called the *Bit-Based Simulated Crossover* (BSC). In this method, the probability for each bit to be a zero or one is calculated. Then these probabilities are used to generate members that, on average, will have the same ratio of ones and zeroes for each bit position that the single point or two point crossover would have generated.

Many other crossover methods are now conventionally being developed.

2.3.3 Mutation

This is a fundamental transformation operator. It creates a new solution in the neighbourhood of the current solution by introducing a small change in some aspect of the current solution. In practise, for example, if binary coding is being used, a single bit in a string may be altered (from 0 to 1, or 1 to 0) with a small probability, creating a new solution. Crossover aims at recombining parts of good substrings from good parent strings to hopefully create a better offspring. Mutation, on the other hand, alters a single child locally to hopefully create a superior child string. The progenies are then evaluated and tested for termination of the algorithm.

ʌlutation alone does not generally advance the search for a solution, but it does provide ısurance against the development of a uniform population as would result for pure crossover nd selection repeatedly applied, incapable of further evolution [25].

'rom the above one may see that the effect of crossover may be detrimental and the good ıformation in the parent strings may be lost. In order to preserve good strings in the opulation and not to allow them to get disrupted by this operator, not all strings are subjected ɔ crossover. A control parameter called crossover probability P_c is set. Subsequently, (1 - $_c$) × 100 % of the strings do not participate in the crossover and are directly copied to the ıating pool. Typical values of P_c are in the range 0. 5 - 1. 0. Bit change by mutation also ccurs, with a probability P_m . Typically P_m is fixed between 0. 005 - 0. 05. Note, however, ıat the values of the probabilities of crossover and mutation are highly problem specific and etermination of optimal values of these is always a challenge. There exist some "rules of ıumb" guidelines to determine these values [13]. Several other approaches to determine the ɔtimum settings of these control parameters also exist, as follows.

ne GA parameterization approach has been suggested by . Patnaik and Srinivas [47]. In this *laptive* probabilities of P_c and P_m have been used to meet the twin goals of maintaining versity in the population and sustaining the convergence capacity of the GA. This method ıs named Adaptive Genetic Algorithm (AGA) because here the probabilities of crossover d mutation are varied depending on the fitness values of the solutions.

'efenstette [23] has formulated the problem of selecting P_c and P_m as an *optimization ɔblem* in itself, and has recommended the use of a second-level (meta-) GA to determine the parameters of the GA. The disadvantage of the Grefenstette method is that it is computationally expensive.

Another approach for optimizing the parameters is the statistical design of experiments approach [33, 40, 41, 44]. This method is both efficient and often very effective. It is used in the present work and is discussed in detail in chapter 4.

If the GA's termination criterion is not met, the population is operated upon by the three GA operators crossover, mutation and selection iteratively. This process is continued till the termination criterion is met. One cycle (iteration) of these operations to produce the offspring is called a *generation* in the GA terminology.

Thus, a genetic algorithm that manipulates a population of few hundred strings actually samples a vastly larger number of regions traversed by decision "hyperplanes" [22]. This implicit parallelism gives the genetic algorithm its central advantage over other problem-solving processes. The genetic algorithm's implicit parallelism allows it to test and exploit large numbers of regions in the search space while manipulating relatively few strings[25]. Implicit parallelism also helps GAs to cope with non-linear problems.

2.4 GAs as Function Optimizers

The standard manner of using GAs as function optimizers is to keep track of the (globally) best individual produced so far regardless of whether that individual exists in the current population [16]. That is, the population is viewed as a database of samples from which potentially better individuals are to be generated. If GAs are applied in this way as function optimizers, how well do they perform relative to other techniques? Experiments have yielded performances ranging from spectacular to "lousy" [16]. The following are the some of the methods contrieved to improve the performance of GAs as function optimizers [16].

2.4.1 Scaling Issues

Although GAs rapidly locate the region in which a global optimum exists, they don't *locate* the optimum with similar speed. A typical solution to this involves providing feedback to the GA in the form of dynamically scaled fitness function in order to maintain sufficient selective pressure between competing individuals in the current population to push the population toward the global optimum [17, 22].

2.4.2 Ranking Approaches

This consists of substituting *rank-proportional* selection for the conventional fitness-proportional selection. By keeping the population sorted by fitness and performing selection on the basis of rank based on fitness, a constant selection differential is maintained between the best and worst individuals in the population. This improves the performance of GA-based function optimizer, but it depends on appropriate use of rank-based selection functions (linear, exponential etc.).

2.4.3 Elitist Strategies

Frequently, performance improvements may be obtained by singling out the best and/or worst individuals in the current population, for according these "elites" special treatment. Examples of this method include "always keeping the best individual found so far in the population" (it only gets replaced by globally better individuals), among other variations.

2.5 Genetic Algorithms Vs Traditional Optimization

Like GAs, some traditional methods such as Box's EVOP [7] method are population-based. But, these methods do not use previously obtained information efficiently. In GAs, previously found good information is emphasized during proportional reproduction and propagated through crossover and mutation operators [4]. Also, GAs permit the capture of *multiple* optimal solutions. Such solutions as we note in chapter 3, can facilitate multi-objective function optimization.

GA operators, unlike gradient-based and other similar methods, do not work deterministically. Almost without exception, traditional optimization methods use fixed transition rules to move from one solution to another. By contrast, in GA, crossover and mutation both use probabilistic transitions (though selection is not purely probabilistic). Thus, the traditional methods may be highly dependent on the starting solution while GAs, when appropriately parameterized, can exhibit a great deal of *robustness* (independence of search performance from starting conditions) [4]. In fact, a properly parameterized GA "run" rarely gets trapped in a locally optimum point. Also, it converges to the global optimum with a high degree of consistency regardless of random selection of crossover site and mutating bit used. Some of these directions may lead to the global basin while others may not. However, the fitness-based reproduction operation indirectly filters good directions while it guides the search. The search in the mutation operator is, however, similar to a local search around a current good solution as done in the local exploratory search in the Hooke-Jeeves method [11].

However, traditional optimization methods can be very effective in solving *special* class of problems. GAs can be rather slow in these applications. For instance, gradient search methods can outperform almost any algorithm in solving *continuous, unimodal* problems even though such methods are not effective with multi-modal problems.

Simple genetic algorithms (SGAs) are at their best when exploring complex landscapes to locate regions of enhanced opportunity. But if a partial solution can be improved further by making small changes in a few variables, it is best to augment the genetic algorithm with other, more standard or conventional search methods [25].

2.6 Advanced Variations of Genetic Algorithms

GAs trace a complex stochastic evolution process [4]. It is also observed that the choice of parameters such as the probability of crossover, the probability of mutation, the nature of the function or response being optimized, the type of crossover and mutation schemes implemented, etc. *interact with each other* in a complex manner to determine the algorithm's convergence. Convergence being a primary concern, a number of different approaches to model the GA process have been recently attempted. The key question, as far as the use of

GAs in practical optimization problem solving is concerned, still seems to remain open: Is the optimal individual solution reliably discovered by GAs in a reasonable time?

Two approaches that would improve a GA seem to emerge here. The first is empirical: Parameterize the GA appropriately to maximize the rate of convergence while ensuring its robust performance This is the approach used in the present work. In this approach statistically designed experiments are used to help parameterize the GA to ensure its fast convergence. The second approach is to use GAs to find a "reasonably good" rather than the globally optimal solution—in a reasonably short time. Mitchell [38] calls this the "satisficing" approach. The GA solution thus produced may be then used as the *starting point* of some conventional optimization method, such as hill climbing. This is thus a *hybrid* method and in many applications it is reported that such a solution strategy performs better than a GA alone (Hart and Belew, 1995).

GAs have also been constructed now to tackle *multi-objective* problems efficiently. It has been shown that GAs can often develop Pareto optimal solutions in such situations successfully [12].

Nevertheless, many other questions about GAs still remains open, such as what is the expected time to find a particular fitness level or what makes a problem hard for GA. Also, a GA can spend considerable amount of time without showing improvement, and then suddenly produce a jump. Such phenomenon cannot be predicted ahead of time. Answers to such questions would enable us to determine what GAs are good at doing or what controls its effectiveness. Two other popular meta-heuristic global optimization methods are simulated annealing [10] and tabu search [21]. Still, considerable evidence has already accumulated about the usefulness of GAs in solving a large variety of real life optimization problems, including engineering design optimization problems [32].

Typical applications of GAs may include optimization of numerical functions of continuous or discrete variables as well as discrete fitness measurements such as those arising in combinatorial optimization e.g. shop scheduling. The simplified description of the GA given in this chapter applies to the optimization of a *single* objective function f(x). As stated earlier, GAs have also been developed to be useful in multi-objective optimization. The following chapters address utility of GAs in multi-objective optimization.

Chapter 3

THE NONDOMINATED SORTING GENETIC ALGORITHM: NSGA

This chapter provides an overview of the nondominated sorting genetic algorithm (NSGA), a multi-objective GA designed by Deb and Srinivas [12] that produces Pareto optimal solutions. The problem of multiple criteria robust design of electronic filter optimized earlier using NSGA [44] is reworked to set the ground for a comparison of the performance of NSGA with ENGA, an improved multi-objective GA developed in this work.

ENGA is presented in the next chapter. The actual comparison of NSGA and ENGA is presented in Chapter 5.

3.1 Multi-Objective Optimization

Many real world decisions involve the simultaneous optimization of multiple objectives[11]. In principle, multi-objective optimization is very different from single objective optimization[12] when the objectives are in conflict. As indicated in Chapter 1, many traditional optimization methods combine the multiple objectives at hand into a single objective by assigning some subjective, *a priori* weights to each objective. But the final solution thus obtained may be very sensitive to small changes in these weight factors[26]. In order to overcome this difficulty the concept of nondomination have been adapted to deal with multiple objectives. The procedure incorporates the economic concept of "Pareto domination" and it does not require use of a priori weights. A key requirement for Pareto domination still is that the objectives in question must be in *conflict* (see Section 1.2).

In single-objective optimization, one's aim is to obtain the single best decision, which is usually the *global* minimum or maximum depending on whether the problem involves minimization or maximization. In multiple objectives, however, there may not exist a unique *single* solution that is globally minimum or maximum with respect to all the objectives. However, in these situations there exists a *set* of solutions that are superior in a *special* way to the rest of solutions in the solution space when all the objectives are considered together. These solutions are known as "Pareto optimal solutions" or "non-dominated solutions". The rest are called dominated solutions (see Figure 1.1). The subsequent selection of one of these solutions by the decision maker from among the Pareto optimal solutions would require domain knowledge e.g. finance, engineering or social welfare, and the consideration of other problem-related factors such as cost, aesthetics, or some other subjective criteria applied to the solutions resident on the Pareto optimal or "efficient front".

Since the GA is a population-based search technique, it was proposed that a *number* of these special solutions be captured in the GA population to subsequently find all possible tradeoffs among the conflicting objectives. These special solutions would be the non-dominated solutions which in the solution space lie on the Pareto optimal frontier. NSGA was among the first such approaches designed.

3.2 The NSGA Algorithm

The idea behind the non-dominated sorting GA procedure is that in it a ranking method is used to emphasize good solutions (those that dominate others in the Pareto sense) and a niche formation method is used to maintain stable population of these good solutions[12]. The working of NSGA differs from the Simple Genetic Algorithm (SGA) only in the manner the *selection* operator works. The other — operators crossover and mutation — remain unchanged from SGA. Before selection is performed in NSGA, the population is ranked on the basis of the non dominated concept (explained above) to achieve Pareto optimality. Figure 3.1 obtained from [12] shows the flow chart of NSGA algorithm.

As the flowchart shows, from the initial population all non-dominated individuals are first identified. These initial non-dominated individuals constitute the first "non-dominated front" in the population and assigned a *dummy fitness value*, equal to population size. This dummy fitness then gives equal reproductive chance to each of these initial non-dominated solutions. To maintain diversity on the front, an operation known as "Sharing" (explained below) of the individuals on the current front is done. Next, these individuals in the first front are completely ignored (put aside) to process and classify the *rest* of the population. In a sequential manner, first, individuals on the *second* front are identified non-dominated as earlier. These individuals are assigned new dummy fitness value (which is kept smaller than the shared dummy fitness value of each of the individuals on the first front). Different dummy

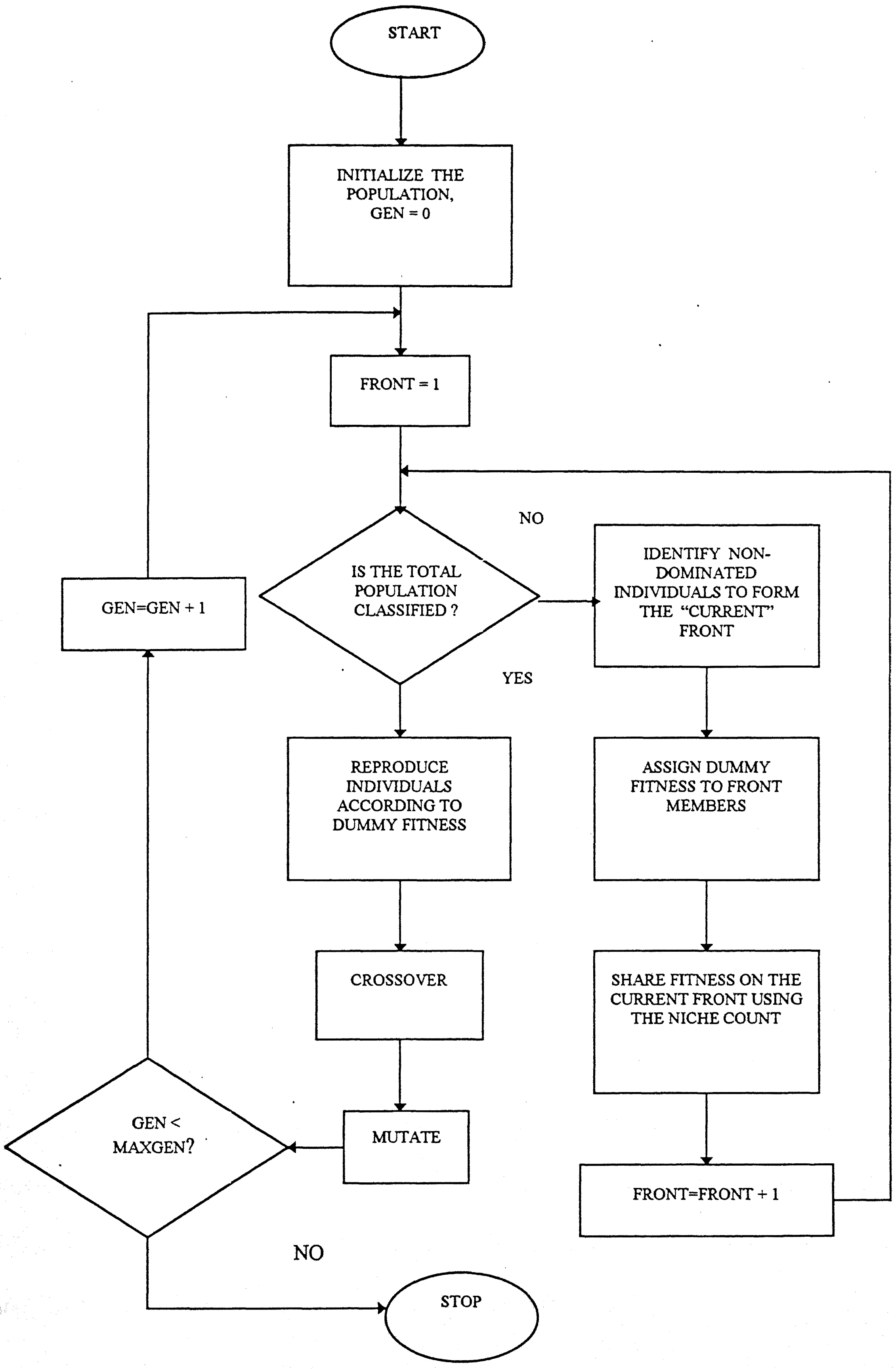
START
INITIALIZE THE POPULATION, GEN = 0
FRONT = 1
IS THE TOTAL POPULATION CLASSIFIED ?
NO
YES
IDENTIFY NON-DOMINATED INDIVIDUALS TO FORM THE "CURRENT" FRONT
ASSIGN DUMMY FITNESS TO FRONT MEMBERS
SHARE FITNESS ON THE CURRENT FRONT USING THE NICHE COUNT
FRONT=FRONT + 1
REPRODUCE INDIVIDUALS ACCORDING TO DUMMY FITNESS
CROSSOVER
MUTATE
GEN < MAXGEN?
GEN=GEN + 1
NO
STOP

Figure 3.1: Flowchart of NSGA

fitness values are assigned to the different fronts (produced by classification) to maintain differentiation between the population of the first front and that of the second (and subsequent ones). Sharing is again done on the second front. This process continues until whole population is classified, which leads to the formation of several consecutive fronts of non-dominated individuals.

After the complete population has been classified, the individuals in the population are reproduced according to their relative dummy fitness values. Now crossover and mutation steps are performed as in SGA.

The complete operation continues until maximum generation (a termination condition) indicated is reached.

3.3 Sharing of Fitness - A Key Concept in NSGA

As already mentioned, NSGA differs from SGA only in the way the selection process works. In NSGA selection is based on fitness obtained after *sharing*. Sharing is done to maintain diversity in the population, a concept borrowed from the sharing of natural resources in a biological habitat leading to *niche formation* in nature to enable diverse types of organic species to co-exist by sharing available resources. The sharing of fitness in NSGA is achieved by first calculating a quantity known as *sharing function value* $Sh(d_{ij})$ between two individuals i and j in the same front, using the following formula as follows[14]

$$\begin{aligned} Sh(d_{ij}) &= 1 - (d_{ij} / \sigma_{share})^2, \quad \text{if } d_{ij} < \sigma_{share} \\ &= 0, \quad \text{otherwise} \end{aligned}$$

where d_{ij} is the phenotypic distance allowed between individuals identified by *i* and *j*, σ_{share} is the maximum phenotypic distance allowed between individuals that would become member of a 'niche' and therefore participate in sharing to coexist. The parameter (d_{ij}) is measured as the (phenotype or real value) difference in the decoded problem variables between two individuals *i* and *j*. (There exists another type of sharing, known as *genotype sharing*, where the sharing function is evaluated using the number of *different coding bits* between the two individuals[14]). Phenotype sharing has been shown to be superior to genotype sharing for the effective operation of NSGA [14].

A *niche* (akin to biological niches formed by organisms by resource sharing) is a stable, non-competing sub-population of individuals surrounding the important peaks in the fitness landscape in the search space. After the sharing function ($Sh(d_{ij})$) has been calculated, the parameter "niche count" is calculated for each individual *i* by adding the sharing function values { $Sh(d_{ij})$ } of this individual with all other individuals {j} coexisting with it on the current non-dominated front. Finally, the shared fitness value for individual i is calculated by dividing its dummy fitness value by its niche count. It is this shared fitness value that is subsequently used for reproduction. The effect of sharing principle depends primarily on the value of the parameter σ_{share}.

Applications of NSGA have gained quick acceptance and now include flow and job shop scheduling problems, routing problems, product design problems and the traveling salesman problem [11].

3.4 Multiple-Criteria Robust Design of Electronic Devices: An Application of NSGA

In this section we first recall the robust engineering design methodology, a prime area requiring multiobjective optimization. We then cite an example.

3.4.1 The Multiple Criteria Robust Design Problem

It is usual for devices such as integrated-circuit operational amplifiers, active and passive filters, to have to satisfy more than one performance objective. For example, for filters, targeted low/high cut-off frequencies as well as quality are often sought simultaneously [3]. The multi-criteria design problem presented in this section was earlier tackled by Sharma [44]. We re-work this problem and then use the results to compare the performances of NSGA with an *alternative* and newer method to evolve Pareto optimal solutions. The material in this section is based on references [3] and [44].

In traditional circuit design, as well as the development of the performance design equations, *sensitivity analysis* of the circuit developed remains a key step that the designer must complete before his job is over. Sensitivity analysis evaluates the likely changes in the device's performance, usually due to element value tolerances or to element value changes with time and temperature.

The approach of *robust design*, by contrast seeks out the *optimum settings* for design parameter values in order that the design becomes robust (i.e., much less sensitive) to variations in the affecting environmental (or other uncontrolled) factors that may cause performance to go off-target. The literature now reports numerous case studies that suggests that, in many situations, conservative designs are unnecessary [6][15][39]. These studies show that robust design can greatly reduce any concern over off-target performance emanating from inadequately controlled manufacturing conditions, temperature/humidity shifts, poor quality components, and also the abuse that final product may suffer because of supply voltage/frequency fluctuations.

Robust design should not be confused with rugged or conservative design, which adds to unit cost by using heavier installation or high reliability, high tolerance components. Robust design seeks to reduce the *sensitivity* of product/process performance to the uncontrolled factors through a careful selection of the values of the design parameters. Taguchi [52] was the first to point out the high engineering and economic value of the robust design.

Taguchi considerably simplified the robust design procedure by adopting the additive or main-factors-only model [30, 52, 8, 6]. In fact, whenever additivity exists, one should approach the optimization problem using what has become known as Taguchi's two-step procedure. But usually in the case of electronic design the interaction effects among various factors are significant or several different performance characteristics must be made robust simultaneously. Thus the two step procedure cannot guarantee the robustness. Taguchi's two step procedure can be described as follows.

Following the notation due to Phadke [39], one may state the robust design problem as follows. Let θ be the set of design parameters that the designer will specify and let x denote the set of noise (uncontrolled) factors. Let $y(x, \theta)$ denote the observed performance haracteristic - for certain particular values of the parameters (x, θ). Let $\mu(\theta)$ and $\sigma^2(\theta)$ be the nean and variance of y, the performance under the influence of noise x. Let the target erformance be μ_0.

One may then state the design optimization problem as:

$$\mathrm{Min}_\theta\ \sigma^2(\theta)$$

Subject to

$$\mu(\theta) = \mu_0$$

Taguchi [52] had suggested that the above problem be solved as an *unconstrained two-step optimization problem*, using a scaling or adjustment factor. In this procedure one design factor θ_1 is postulated to be an adjustment factor. The response characteristic then splits up into $g(\theta_1)h(x, \theta^1)$, $h(x, \theta^1)$ being independent of θ_1. The optimization procedure then becomes:

Step 1: Choose θ^1 such that it minimizes

$$\frac{\sigma^2{}_h(\theta^1)}{\mu^2{}_h(\theta^1)}$$

Step 2: Choose θ_1 such that

$$\mu_h{}^2(\theta^1) = \mu_0$$

Phadke[39] suggests that the adjustment factor θ_1 should be identified by experimentally examining the effect of all design factors on the signal-to-noise (S/N) ratio and the mean μ. Any factor that has no effect on the S/N ratio - defined as $\log_{10}[\mu_h{}^2(\theta^1) / \theta_h{}^2(\theta^1)]$ - but a significant effect on μ, can be used as the adjustment factor. As noted in several real-life design situations, however, this identification isn't always clear cut, especially when the relationship between σ_h and μ_h is complex [8], or when the effects interact [19]. Taguchi has not offered any specific guidelines on what further analysis and experiments should be done when the 'verification experiment' fails to support the main-factors-only assumption [20]. One example of a multiple-criteria design problem that also contains interactions is the design of a passive filter (Figure 1) described in detail in Flippone [19] and Suh [48], in which two target performance requirements must be met, stated respectively as:

$$\text{Cut-off frequency } \omega_c = \frac{(R_2 + R_g)(R_s + R_3) + R_3 R_s}{2\pi(R_2 + R_g)R_3 R_s C} \tag{1}$$

$$= 6.84 \text{ Hz}$$

and

$$\text{Deflection} \quad D = \frac{|Vo|}{G_{sen}} = \frac{|V_s| R_g R_s}{G_{sen}[(R_2 + R_g)(R_s + R_3) + R_3 R_s]} \tag{2}$$

$$= 3.00 \text{ in.}$$

where D is deflection, G_{sen} is the galvanometer sensitivity in mV in^{-1}.

The above problem is solved using an alternative design method, also empirical as Taguchi's two-step optimization method is, two make the design robust.

3.4.2 Target Performance Requirements as Explicit Constraints

One must realize that when one searches the design space for optimum values for the design parameters θ, not all design parameters in θ are free to take any value in the real space. The *permissible* values for these parameters are constrained by the requirement that each performance characteristic y must be on target when the design is complete.

Multi-objective robust design problems such as the passive filter problem above may be tackled explicitly as *constrained optimization problems*. There are some distinctions between this approach and the two-step procedure. In the two-step procedure, an adjustment parameter is employed to adjust performance to target after the appropriate S/N ratio has been maximized with the help of control parameters (Taguchi, 1986; Phadke, 1989). by contrast, the constrained approach would not require one to identify an adjustment parameter. This becomes a significant convenience because, in general, performance (the responses to be made robust) may depend on more than one design parameter and also perhaps on their interactions, in complex ways. Such is the case when the passive filter, as indicated by the equations 1 and 2 for ω_c and D respectively, the **ANOVA** tables given in Flippone [19] and the interaction effects detectable by an L_8 Orthogonal experiment.

3.4.3 Rationale for a Constrained Robust Design Approach

The primary difficulties in applying Taguchi's two-step approach to certain design problems are as follows.

1. Interactions are mostly ignored, to be included later in an expanded inner array if the verification experiment fails to prove satisfactory.
2. When the design parameters are continuous, the approach does not fully utilize the structure of the problem.
3. Transformations aimed at stabilizing the variance of response when the variance depends upon the mean, to permit adjustment of mean to the target (Dehnad, 1989, p. 291) - after one has maximized the S/N ratio do not necessarily make up for disadvantages (1) and (2).

If an alternative procedure is adopted in which the designer restricts his search for a robust design among possible alternative designs, in each of which on-target performance is guaranteed, then as references [3] shows, disadvantage (3) disappears. Further, if appropriate designs and mechanistic or regression models are employed, disadvantages (1) and (2) also disappear.

Reference [3] formalizes the constrained robust design approach as follows. At the outset, it is assumed that the designer knows what the different design parameters are and the performance(s) he is attempting to optimize. Let the best performance desired for characteristic Y_i be identified by a target value τ_i. The objective is to choose design parameter values that will reduce the sensitivity of performance termed noise. The designer should proceed as follows.

Step 1: For each performance characteristic Y_i to be made robust while also to be set equal to some target value τ_i, establish a *quantitative model* relating all the design parameters $\{DP_1, DP_2, DP_3, \ldots\}$ to the performance Y_i as in:

$$Y_i = f(DP_1, DP_2, DP_3, \ldots)$$

Step 2: Write the quantitative model obtained in Step 1 as a *constraint*:

$$f(DP_1, DP_2, DP_3, \ldots) = \tau_i$$

If n different performances $(Y_1, Y_2, Y_3, \ldots, Y_n)$ are being simultaneously targeted (to τ_1, τ_2, τ_3, etc.), then one would establish here a set of n constraints given by equations such as:

$$f(DP_1, DP_2, DP_3, \ldots) = \tau_1$$

$$g(DP_1, DP_2, DP_3, \ldots) = \tau_2$$

$h(DP_1, DP_2, DP_3, \ldots) = \tau_3$

tep 3: Solve the equations formed in Step 2 to obtain certain 'dependent' design parameters in terms of the truly 'independent' design parameters. Clearly, if there are m design parameters and n constraints with m>n, one has the choice of treating m-n design parameters as independent (usable as the robustness-seeking variables in Step 5 below) while the others are dependent (their values are restricted so that the design reaches the desired performance targets).

tep 4: Construct the true inner array using only the truly independent design parameters. Also set up the appropriate outer array, or a Monte Carlo experimental set-up, or the physical arrangements, for repeated observations of performance under the influence of noise. Conduct experiments at each setting of the inner array row to observe performance(s) (Y_1, Y_2, Y_3,...) under the influence of noise.

tep 5: Apply search, response surface methods (**RSM**) [37], or some other technique to find the combination of the independent design parameters that *minimize* the empirically estimated variance (of each performance Y_1, Y_2, Y_3, etc.) under the influence of noise. If a unique set of design parameter values does not optimize all performance characteristics, develop Pareto-optimal set of candidate designs.

3.4.4 Application of Constrained Optimization to the Filter Design Problem

Step 1 stated above establishes mathematical relationship(s) relating the performance characteristic(s) of interest to the design parameters. For the passive filter the application of Kirchoff's laws allows us to derive these relationships (Equations (1) and (2)). Therefore, for assive filter problem, one may directly go to Step 2. As reference [3] shows Equations (1) nd (2) are the results of completing Step 2, which help to constrain the total design space onsisting of all possible values of R_3, R_2 and C to the *feasible* set of solutions for which on-arget performance (ω_c = 6.84 Hz and D = 3 in) would be *fait accompli.* Given ω_c (+6.84 Iz), D(= 3.00 in), and a value of R_3, the two remaining design parameters (R_2 and C) may be ıbtained as:

$$R_2 = \frac{|V_s|R_g R_s - DG_{sen}R_3R_s}{DG_{sen}(R_3 + R_s)} - R_g \qquad (3)$$

ınd

$$C = \frac{|V_s|R_g(R_3 + R_s)}{2\pi\ \varpi_c R_3(|V_s|R_g R_s - DG_{sen}R_3R_s)} \qquad (4)$$

The combination of R_3, R_2 and C thus obtained (with D = 3 in and ω_c = 6.84 Hz) is a *feasible* lesign (though not yet robust or optimum), because it satisfies the target ω_c and D equirements (1) and (2). The next step is to search for a feasible design (here a value of R_3) vhich maximizes robustness. Equations (3) and (4) together define the contour on which one ieeds to conduct this search. Consideration of non-linear constraints in optimization problems s well known.

Use of Lagrange's multipliers when the analytical form of the objective function is known constitutes a standard constrained optimization procedure (Nemhouser *et al.*, 1989). In the present situation, however, the exact mathematical dependence of the S/N ratios on the design parameters is unknown; hence a (constrained) 'search' was deemed by [3] as an acceptable practical approach.

ne realizes Step 3 by writing Equations (3) and (4). It is clear that the filter design problem uly has only *one* design parameter, which here is R_3, the sole independent design parameter; nce we select a value of R_3, the meaningful (feasible) values of R_2 and C become known ixed) by (3) and (4).

he final choice of R_3 would depend on resolving the multiple (two-) objective problem.: (1) aximize the robustness of cut-off frequency ω_c, and also (2) maximize the robustness of eflection D. One may now identify a set of Pareto-optimal designs (each such design is better ian any other design possible, at least with respect to one performance criterion). Other riteria may be then applied to select the "best" filter design among these Pareto-optimal esigns. The two extreme members of the Pareto-optimal set of designs here appear to be:

$R_3 = 300,\ \ R_2 = 29,\ \ \ C = 454.4\mu F$

:his design maximizes $S/N_{\omega c}$) and

$R_3 = 200,\ \ R_2 = 106,\ \ C = 424.1\mu F$

:his design maximizes S/N_D).

)ne may find acceptable, Pareto-optimal designs, if necessary, by search, or by enumeration, etween these two extreme designs.

}.5 Application of NSGA to Electronic Filter Design

harma [44] originally applied NSGA to the electronic filter problem. In this method two bjectives are considered for simultaneous minimization. The first one is the variance of cut- ff frequency (ω_c) and the second is the variance of deflection (D). Resistance R_3 is used as the only control variable. An L_8 orthogonal array (OA) is used to "create" noise in the parameters [2] to simulate variability due to inexpensive components to be used in fabricating the filter.

Table 3.1 gives the noise conditions used to form the L_8 noise array.

Table 3.1: Tolerance levels causing noise

	Level		
Components	**1**	**2**	**3**
$R_3(\Omega)$	R_3 - 5%	R_3	R_3 + 5%
$R_2(\Omega)$	R_2 - 5%	R_2	R_2 + 5%
$C(\mu F)$	C - 5%	C	C + 5%
$R_s(\Omega)$	119.82	120	120.18
$R_g(\Omega)$	97.853	98	98.147
$G_{sen}(\mu V/in)$	656.594	657.58	658.566
$V_s(V)$	0.014978	0.015	0.015023

The two objective functions considered for optimization are

$$f_1 = \text{Min}(\text{Var\_}\omega_c)$$

$$f_2 = \text{Min}(\text{Var\_D})$$

The above two variances are calculated from the L_8 orthogonal array [2] as follows.

For each chromosome generated by GA, the value of R_3 may be found by decoding it. Then for this value of R_3 the corresponding values of R_2 and C can be found from the equations (3) and (4).

Table 3.2: L_8 orthogonal array used in the study

Expt.	R_3	R_2	C	R_s	R_g	G_{sen}	V_s	ω_c(obs)	D(obs)
1	–	–	–	–	–	–	–		
2	–	–	–	+	+	+	+		
3	–	+	+	–	–	+	+		
4	–	+	+	+	+	–	–		
5	+	–	+	–	+	–	+		
6	+	–	+	+	–	+	–		
7	+	+	–	–	+	+	–		
8	+	+	–	+	–	–	+		

The noise (outer) OA (Table 3.2)is constructed using the tolerances mentioned in the Table 3.1. For each of the eight "outer array" experiments the values of the cut-off frequency (ω_c) and deflection (D) are found from the equations (1) and (2). The mean for the eight experiments and the variance for the two parameters are calculated as follows.

$$\omega_{c,mean} = \sum_{i=1}^{8} \omega_{c(obs),i}$$

$$D_{mean} = \sum_{i=1}^{8} D_{(obs),i}$$

$$\text{Var}(\omega_c) = f_1 = \frac{1}{(8-1)}\sum_{i=1}^{8}\left[\omega_{c(obs),i} - \omega_{c,mean}\right]^2$$

$$\text{Var}(D) = f_2 = \frac{1}{(8-1)}\sum_{i=1}^{8}\left[D_{(obs),i} - D_{mean}\right]^2$$

Since both the objective functions (variances) are to be minimized to maximize the filter's robustness when subject to noise, the fitness values for the chromosome are calculated as

$$F_1 = \frac{1.0}{(1.0+f_1)}$$

$$F_2 = \frac{1.0}{(1.0+f_2)}$$

Similarly for all the members of the GA population in a generation the two fitness values are calculated. The subsequent operations (i.e. selection, crossover and mutation) create progenies for the next generation. This procedure terminates when desired convergence criteria is met.

3.5.1 Parameters used in NSGA Execution

• *Population size*	=	100
• *Maxgen*	=	50
• *Pcross*(P_c)	=	0.8
• *Pmute*(P_m)	=	0.01
• *Lchrom*	=	32
• R_3 *(min)*	=	1
• R_3 *(max)*	=	350
• *Dshare*	=	0.1

The Figures 5.8 and 5.10 show the initial population of chromosomes and the Pareto obtained after 10th generation. From these two figures we can observe the movement of the members as NSGA executes on to the nondominated (Pareto) front after they are randomly generated. Subsequently from the resulting solutions (Table 5.16) on the Pareto front the decision maker will be expected to select any solution depending his preference criteria and performance requirements. A particular specification of the two variances by the decision maker yields the corresponding values of the resistances R_3, R_2 and capacitance C to be used in the electronic filter, which makes the filter's performance robust.

3.6 Conclusions

n this chapter the nondominated sorting genetic algorithm (NSGA), a heuristic method for the ›ptimization of multiple objectives was recapitulated briefly. An electronic filter problem ›ptimized earlier using NSGA considering two objectives (cutoff frequency and the leflection) to produce Pareto optimal solutions was reworked and illustrative results were ›resented. The succeeding chapter describes *ENGA*, a new algorithm developed in this work or numerical multiobjective problems and then a comparison of performance of NSGA with ENGA.

Chapter 4

ENGA: AN ELITIST NON-DOMINATED SORTING GENETIC ALGORITHM

This chapter describes a new nondominated sorting genetic algorithm (that we have called ENGA) and it addresses the selection of optimal parameters for ENGA to achieve its fast convergence. Chapter 5 compares NSGA with ENGA to evaluate their relative efficacy in finding Pareto optimal solutions.

4.1 ENGA : A Key Enhancement of NSGA

The enhancement made in NSGA in the present work to evolve ENGA (the *Elitist* Nondominated Sorting GA) based on suggestions by Bagchi [5] was motivated by the fact that NSGA does not necessarily preserve good solutions from generation to generation. A collection of good (near optimal) solutions may in fact get destroyed and its members may reappear in future only probabilistically. By contrast, ENGA introduces a key preservation feature, as follows.

igure 4.1 outlines the steps in ENGA and it indicates the key enhancement introduced in ISGA to develop ENGA. ENGA does not discard the old population and replace it ompletely by progenies. It produces the progenies through crossover and mutation (equal in umber to parents) and then *selects* the members to constitute the next generation by erforming an additional nondominated sorting of *combined* (parents + progenies) pool. The op ranking 50% individuals of this combined pool are then selected to propagate their haracteristics and these may contain several among the parent chromosomes without making ny modification to them if they are good enough to be able to outrank some of the newly reated progenies. The skimming of the top 50% of the members of the combined (parents + rogeny) pool to construct the subsequent generation makes the ENGA "elitist".

.2 A Bi-objective Test Problem

bi-objective optimization problem originally solved using NSGA by Deb & Srinivas [12] is sed below to illustrate solution by ENGA. Two conflicting objectives are to be minimized ere simultaneously.

he problem :

bjective 1 *Minimize* $f_1(x) = x^2$

bjective 2 *Minimize* $f_2(x) = (x-2)^2$

Subject to $-10 \leq x \leq 10$

he Pareto optimal front for the above problem is the range $0 \leq x \leq 2$ [12], the limits of both e objective functions in this range being 0 and 4. Figure 4.2 shows the theoretically btainable Pareto front for this problem

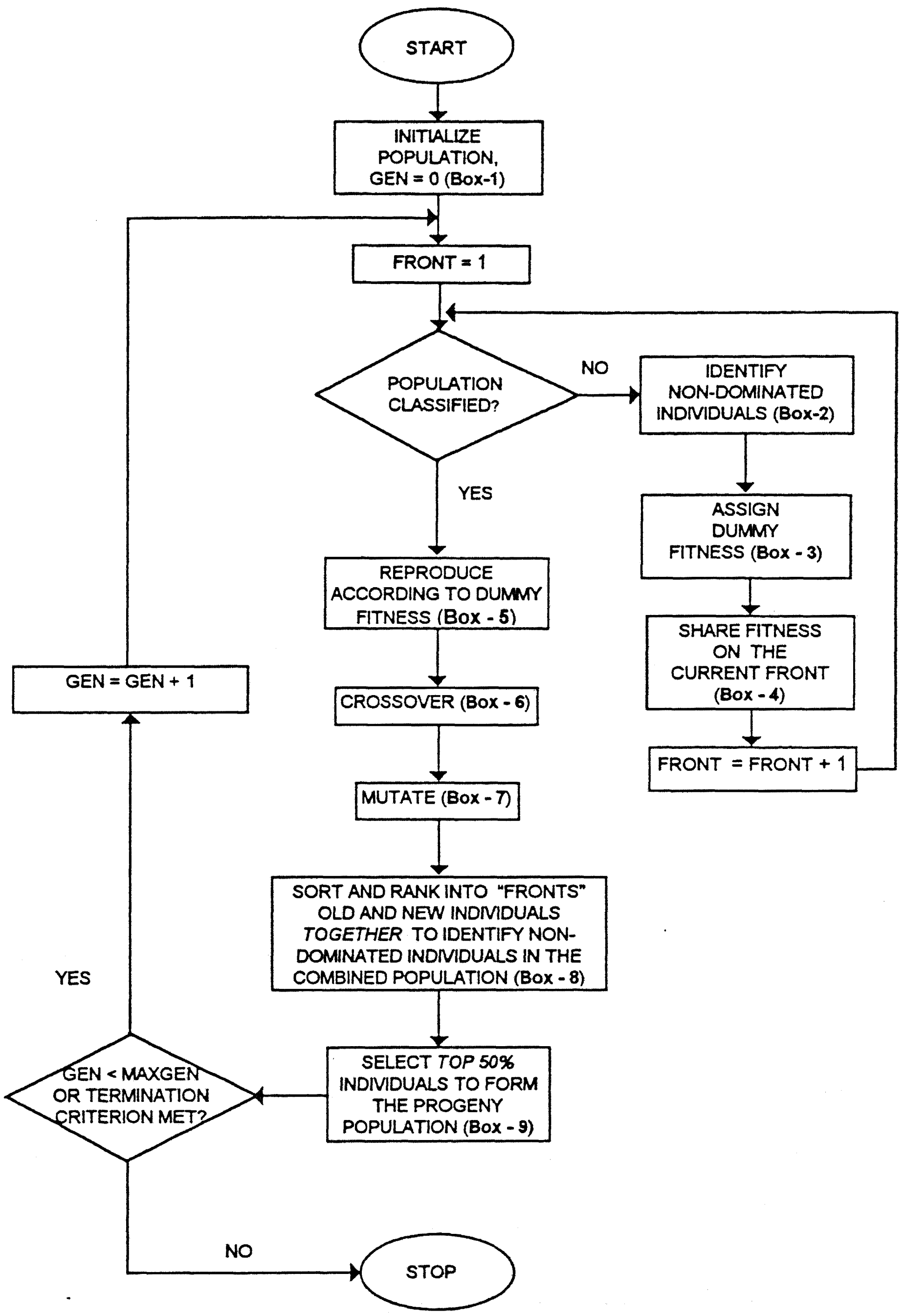
START
INITIALIZE POPULATION, GEN = 0 (Box-1)
FRONT = 1
POPULATION CLASSIFIED?
NO
IDENTIFY NON-DOMINATED INDIVIDUALS (Box-2)
ASSIGN DUMMY FITNESS (Box - 3)
SHARE FITNESS ON THE CURRENT FRONT (Box - 4)
FRONT = FRONT + 1
YES
REPRODUCE ACCORDING TO DUMMY FITNESS (Box - 5)
CROSSOVER (Box - 6)
MUTATE (Box - 7)
SORT AND RANK INTO "FRONTS" OLD AND NEW INDIVIDUALS TOGETHER TO IDENTIFY NON-DOMINATED INDIVIDUALS IN THE COMBINED POPULATION (Box - 8)
SELECT TOP 50% INDIVIDUALS TO FORM THE PROGENY POPULATION (Box - 9)
GEN = GEN + 1
YES
GEN < MAXGEN OR TERMINATION CRITERION MET?
NO
STOP

Figure 4.1: ENGA: The Elitist Non-dominated GA

In the following pages this bi-criteria problem is used as the test bed to determine the optimum GA parameter values for ENGA. The methodology uses statistical design of experiments (DOE), the experiment itself being a 2^4 full factorial design.

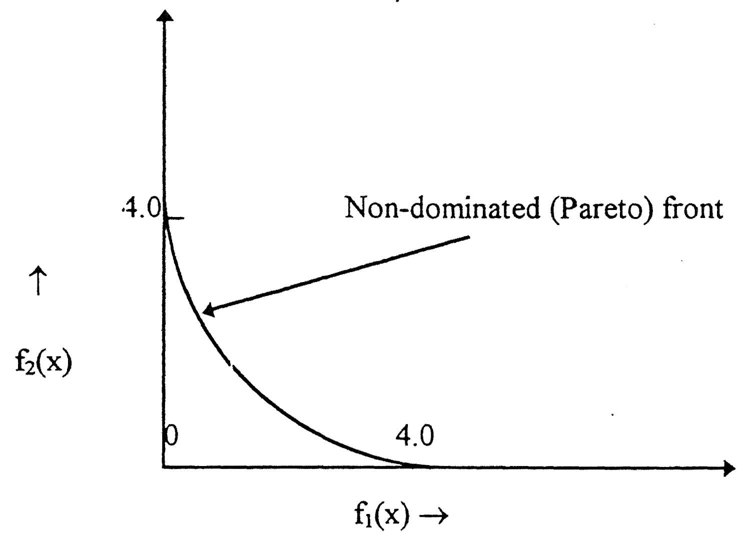

Figure 4.2: The Pareto Optimal (or Non-dominated) Front for the *Bi-objective Problem*.

4.3 Optimization of ENGA Parameters

A key challenge in successfully employing genetic algorithms (and this includes the simple genetic algorithm or SGA and the NSGA, and the ENGA being introduced in this chapter) is the correctness in the choice of the different parameters that control the effectiveness of the algorithm. It is noted, the GA process is a directed search with stochastic sampling occurring in each generation. The effectiveness of this search is greatly affected by the choice we make for the mutation and crossover probabilities P_c and P_m, the population size P_S, and for the NSGA the parameter known as sigma-share (σ_{Share}), which controls the precision with which the Pareto optimal front is combed for the existence of a possible efficient point or "peak" [12].

The GA thus being a "process" influenced by multiple factors and their interactions, we chose to use the framework of statistical design of experiments [37] to discover the "optimal" settings for the different GA parameters. Table 4.2 is an illustration of typical results obtained from such parameterization studies guided by DOE. The four factors considered were as follows.

4.3.1 Population Size(P_s)

The population size effects both the ultimate performance and efficiency of GAs. GAs generally do poorly with very small populations[23], because small populations provide insufficient sample size for most hyperplanes. A large population is more likely to contain representatives from the entire search space. Hence Genetic Algorithms can perform a more informed search. As a result, a large population discourages premature convergence to suboptimal solutions. On the other hand, a large population requires more evaluations per generation, resulting in slow rate of convergence.

4.3.2 Crossover Probability (P_c)

The crossover probability controls the rate with which the crossover of alleles of two parent members of the population takes place. The higher the crossover rate, the more quickly new structures are introduced into the population. If the crossover rate is too high, it discards the better members even if reproduction produces members of improved fitness. If the crossover rate is too low the search may stagnate due to lower rate of exploration of the solution space.

3.3 Mutation Probability (P_m)

[utation helps in local search. A small mutation probability serves well to prevent any given t from remaining forever unaltered to a single value in the entire GA run. A high level of utation on the other hand, yields an essentially random search and thus disruptive to the A's intended "intelligent" search.

.3.4 Sigma Share (σ_{share})

'he performance of nondominated sorting utilizing niche formation and ranking used in NGA depends critically upon the value of sigma- share. A low value of sigma-share ssentially results in search space being narrow. A high value of the same allows broadening f the search. The value of sigma-share also effects the precision of finding distinct solutions n the Pareto front. Somewhere in between would lie the optimum value of σ_{share}.

'able 4.1 shows the different factors and their levels used in the parameterization study.

Table 4.1: Factors and Their Levels Used in DOE for the Test Problem

FACTOR	LEVELS (Lo - Hi)
Crossover probability (P_c)	0.5 - 0.9
Mutation probability (P_m)	0.01 - 0.05
Population size (P_s)	50 - 100
Dshare (σ_s)	0.1 - 5.0

.4 Parameterization by DOE

he experiments were run using the 2^4 full factorial design [37, 2] shown in Table 4.2. Each xperiment started with identical random seeds initializing ENGA and the ENGA running niformly for 250 generations for the parameter combinations indicated by a "row" in Table .2. After the runs terminated, the final solutions were pooled together and subjected to non-ominated sorting, to appraise which parameter setting discovered the maximum number of istinct solutions on the Pareto optimal front. Thus the response for each experiment was aken as the number of distinct solutions on the nondominated front. These counts are shown n Column 6 of Table 4.2. The factor effects themselves are displayed in Figure 4.5, indicating ne relatively high impact of P_s on the transition of solutions to the Pareto front with σ_{share}, P_m nd to a lesser degree P_c contributing to the speed of such movements.

Table 4.2: Results of a 2^4 Full Factorial ENGA Parameterization Experiment

EXPT#	σ_{share}	P_C	P_M	PS	Number of Unique Solutions on the Pareto Front	Percent of Final Population on the Pareto Front
1	0.1	0.5	0.01	50	24	48.0%
2	0.1	0.9	0.01	50	30	60.0%
3	0.1	0.5	0.05	50	37	74.0%
4	0.1	0.9	0.05	50	30	60.0%
5	0.1	0.5	0.01	100	58	58.0%
6	0.1	0.9	0.01	100	57	57.0%
7	0.1	0.5	0.05	100	65	65.0%
8	0.1	0.9	0.05	100	70	70.0%
9	5	0.5	0.01	50	15	30.0%
10	5	0.9	0.01	50	20	40.0%
11	5	0.5	0.05	50	32	64.0%
12	5	0.9	0.05	50	33	66.0%
13	5	0.5	0.01	100	33	33.0%
14	5	0.9	0.01	100	60	60.0%
15	5	0.5	0.05	100	64	64.0%
16	5	0.9	0.05	100	56	56.0%

Table 4.3 shows the parameter values that were found to be optimum based on these full factorial experiments. These values implemented together would enable ENGA to converge to the Pareto optimal front the Fastest.

Table 4.3: Optimum Parameter Settings by DOE for the Test Problem

Parameter	Value
Crossover probability (P_c)	0.9
Mutation probability (P_m)	0.05
Population size (P_s)	100
Dshare (σ_{share})	0.1

With the parameter settings shown in Table 4.3 implemented it was found that ENGA produced a population that had every member on Front 1 (the efficient front) after only the 4th generation. Further execution of ENGA would push down the efficient front toward even lower values of f_1 and f_2. Figures 4.3 shows the 100 initial randomly generated solutions to the problem, indicating the clearly wide scatter of the initial solution population and there being rather few Pareto optimal solutions among them. Figure 4.4 shows the members discovered on the Pareto front after 250 iterations of ENGA when the best parameter combination indicated by on Table 4.3 was used. (This example is typical of the experience with the use of DOE in parameterizing GAs as tested on a fairly large variety and sample of uni- and multi-variate optimization problems[33, 40, 41, 44]). We observed from the DOE study that the parameter population size (P_s) has considerable effect and that a larger population produces numerically more *distinct* Pareto solutions. Next to it is the sigma share (σ_{share}). A low value of σ_{share} resulted in producing larger number of solutions on Front 1.

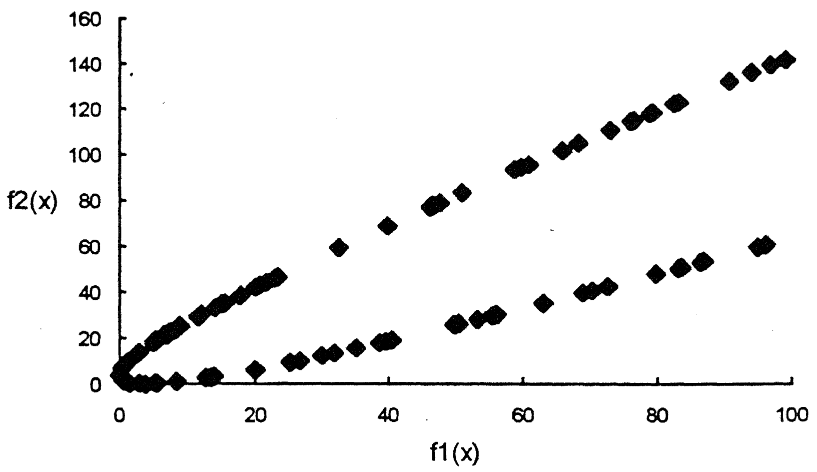

Figure 4.3: 100 Randomly Generated Initial Solutions

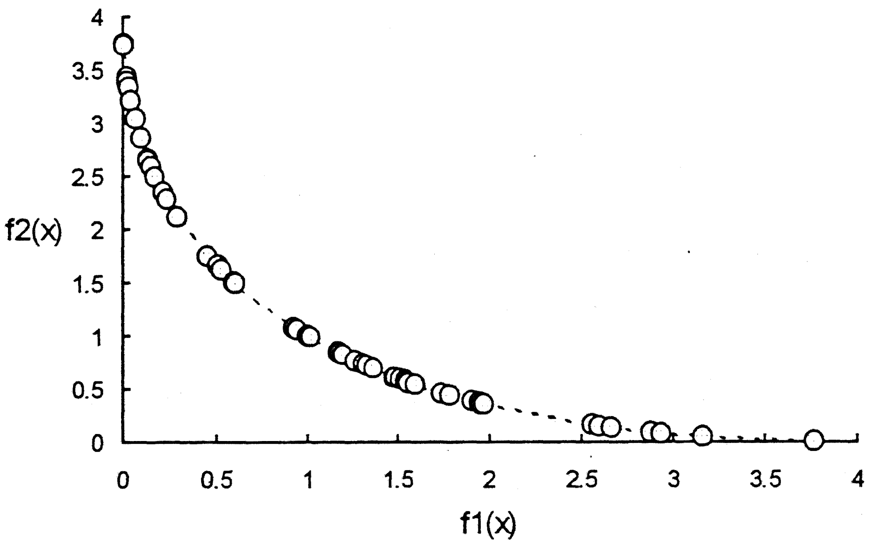

Figure 4.4: Seventy Unique Pareto Optimal Solutions found by ENGA

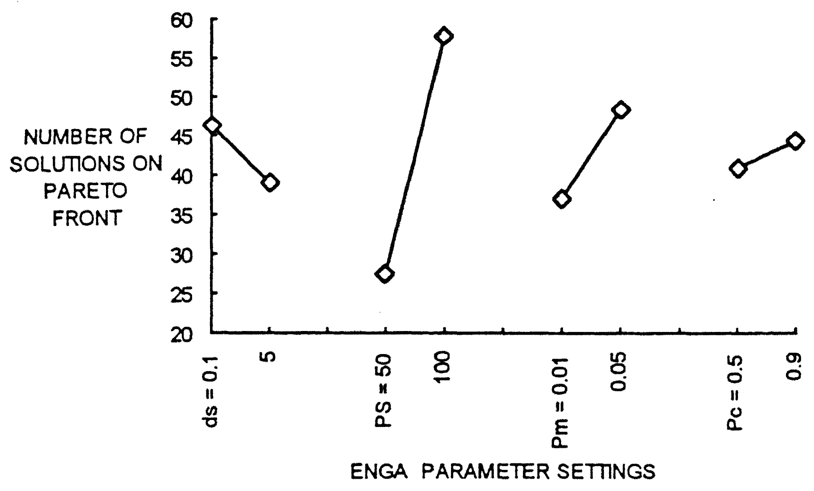

Figure 4.5: Factor Effects on the Discovery of Pareto Optimal Solutions by ENGA

ł.5 Interaction Between Parameter Effects

ιs of today no literature is available to indicate the combined or interacting effects of two (or ıore) parameters on the performance of NSGA. Literature suggests here the conduction of ne-factor-at-a-time studies [23]. Indeed, for parameterization, conventionally only *one* factor ; considered at a time and based on its effect noted a *low* value or a *high* value of it is ubsequently selected as the "best" parameter setting. In the present study using DOE we aried several parameters simultaneously to find their combined effect and to see the extent of ıteraction between them. It became very clear that there exists considerable interaction mong all four parameters, besides single-factor effects (Table 4.4 shows the magnitudes of hese effects—the low and high values are calculated as the average count of nondominated ıembers). Figure 4.6 shows the individual factor effects and interactions on the same scale. t is obvious, therefore, that GA parameters must be themselves carefully optimized, *together*, ing a DOE-type multi-factor based pilot study before the GA is used in a "production" ode.

Table 4.4: Magnitudes of Individual and Interaction Effects of the Four Parameters for the Test Problem

Factor(s)	Low Value	High Value
Dshare (σ_s)	46.375	39.125
Population size (P_s)	27.625	57.875
Mutation Probability (P_m)	37.125	48.375
Crossover Probability (P_c)	41.00	44.50
σ_s - P_c	44.13	41.38
σ_s - P_m	44.25	41.25
σ_s - P_s	41.75	43.75
P_s - P_m	43.00	42.50
P_s - P_c	43.88	41.63
P_m - P_c	. 39.88	45.63

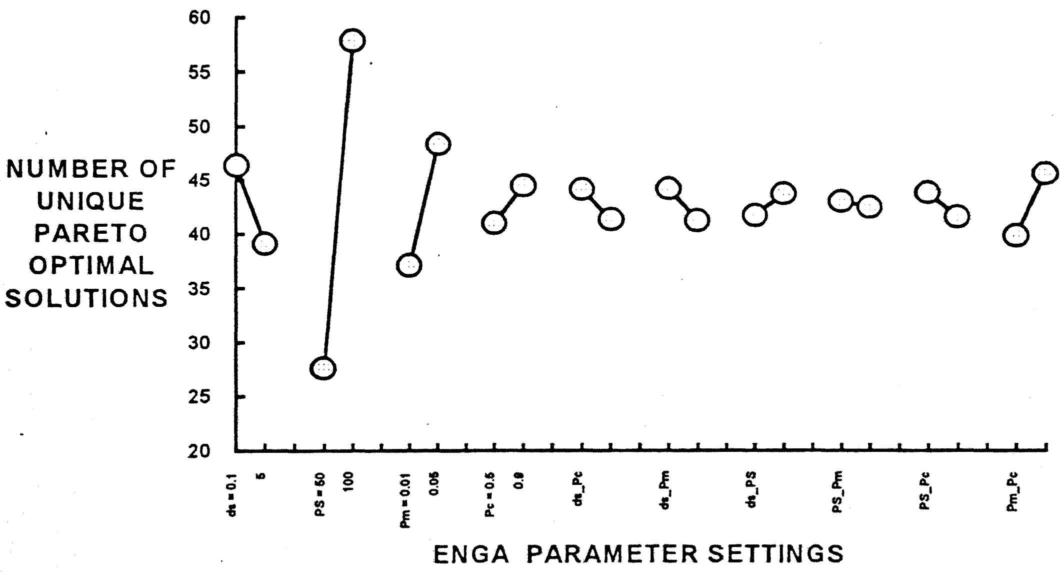

Figure 4.6: Interactions Between Various Parameters

.6 Chapter Summary

ne chapter has described newly developed nondominated sorting algorithm ENGA (an ıhanced version of NSGA) based on Bagchi [5]. Successful parameterization of ENGA was emonstrated using the statistical design-of-experiments framework. Full factorial cperiments were used for this purpose, incorporating the four factors population size, ossover probability, mutation probability and sigma-share. The individual factor effects and everal interactions among the factors were found to be considerable. Guidelines to determine ıe optimum settings of the parameters (P_c, P_m, PS and σ_{share}) in for ENGA to ensure its fast ıd consistent convergence were given.

;hapter 5

;OMPARISON OF NSGA AND ENGA

ı this chapter the two multiobjective genetic algorithms (NSGA and ENGA) are compared ır their relative performance using, two different problems as test beds. The first problem is two objective minimization problem solved earlier by Deb and Srinivas [12] and the second the engineering design (electronic filter) optimization problem described in Chapter 3. The ɔmparison is done on the basis of two different performance criteria, namely, (a) the *number f function evaluations* (a metric indicating the needed computational effort) to produce a iven number of distinct Pareto optimal non-dominated solutions, and (b) the *number* of areto optimal *solutions* discovered in a given number of GA generations. The performances f NSGA and ENGA are compared statistically in each case.

i.1 Test Problem 1: A Bi-objective Optimization Problem

'his problem was originally solved by Deb and Srinivas [12]. It involves two objectives $f_1(x)$ nd $f_2(x)$ and a single decision variable x.

finimize $f_1(x) = -x$ if $x \leq 1$

$= -2 + x$ if $1 < x \leq 3$

$= 4 - x$ if $3 < x \leq 4$

$= -4 + x$ if $x > 4$

Minimize $f_2(x) = (x-5)^2$

.1.1 Criteria 1 : Comparison on the Basis of Number of Function Evaluations

n this criterion the algorithms are compared with respect to the number of function valuations required to meet some pre-set performance requirements. Obviously the algorithm hat takes fewer number of function evaluations would be termed better. The pre-set criterion ısed here is that out of whole population, 95% or more should be on the nondominated front .e., on Front 1. Tables 5.1 to 5.5 show the GA parameter values used for both NSGA and ENGA and the number of function evaluations required by NSGA and ENGA to solve the wo test problems. Each comparison is replicated using five different starting conditions (i.e., nitial seeds).

Table 5.1: Parameters Used in Test Problem 1

Parameter	Value
Population size	100
Pcross	0.8
Pmute	0.01
Length of chromosome	32
Dshare	0.2
X_{min}	-10
X_{max}	10

Table 5.2: Number of Function Evaluations Needed for Test Problem 1 (NSGA & ENGA) to Obtain 95% or More Members on the Non-dominated Front 1

Seed used	Total Function Evaluations for NSGA	Total Function Evaluations for ENGA
760	800	600
7494	3600	600
1835	1600	600
1520	3400	400
1234	6200	600
18	4400	600
1231	8700	500
35	3300	600
1997	1300	600
1000	3100	400

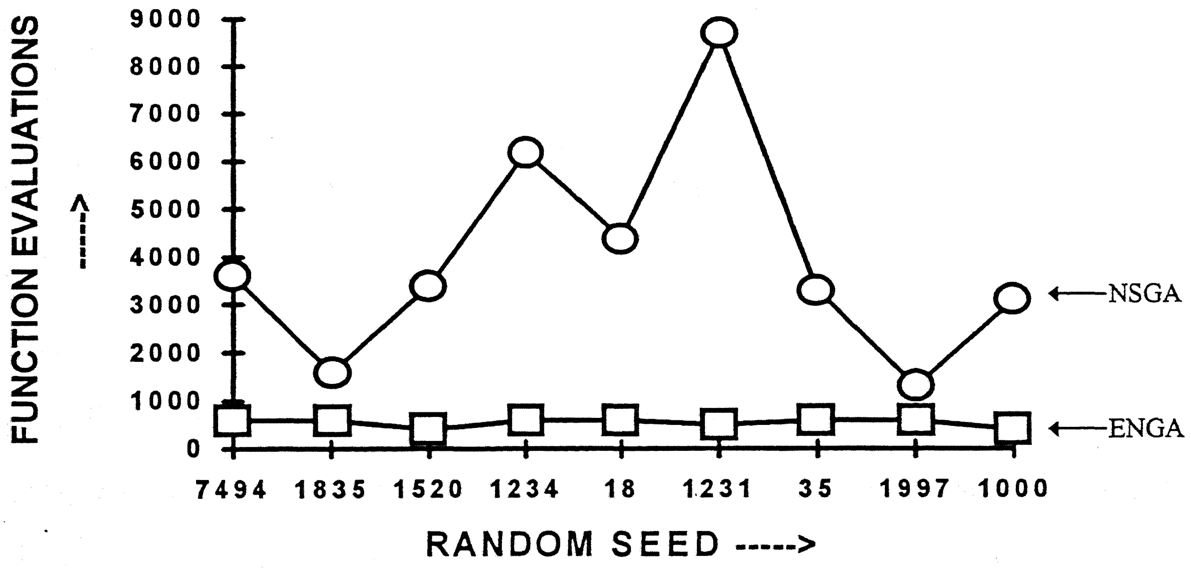

Figure 5.1: Comparison of NSGA - ENGA on Function Evaluations for Test Problem 1

As Table 5.2 indicates, ENGA took considerably fewer function evaluations to deliver 95 % or more distinct solutions on the Pareto front (Figure 5.1 shows the results graphically) . Another point noted is the consistency of ENGA in finding many Pareto optimal solutions irrespective of the random starting conditions. ENGA took almost consistent number of evaluations to meet the pre-set performance target of *finding 95% efficient solutions*. On the other hand NSGA not only required more function evaluations than ENGA but it also appears to be sensitive to the starting conditions (Figure 5.1) .

5.1.2 Statistical Test for Criteria 1: Number of Function Evaluations

The procedure used here is the *Wilcoxon signed-rank test* [35]. This can be applied to the paired data. Let (X_{1j}, X_{2j}), $j = 1,2,\ldots,n$ be a collection of paired observations from two continuous distributions that differ only with respect to their means. (It is not necessary that the distributions of X_1 and X_2 be symmetric). This assures that the distribution of the differences $D_j = X_{1j} - X_{2j}$ is continuous and symmetric. Thus the null hypothesis is H_o: $\mu_1 = \mu_2$, which is equivalent to H_o: $\mu_D = 0$.

To use the Wilcoxon signed-rank test, the differences of function evaluations of both NSGA and ENGA were first ranked in ascending order of their absolute values, and then the ranks were given the signs of the differences. Ties were assigned average ranks. The procedure used was as follows.

Let W^+ be the sum of the positive ranks, W^- be the absolute value of the sum of the negative ranks (shown in Table 5.3), and $W = \min(W^+, W^-)$. The one-sided Wilcoxon signed rank test ests H_o: $\mu_1 = \mu_2$ against the alternative H_1: $\mu_1 < \mu_2$ (or H_1: $\mu_D < 0$). The null hypothesis H_0 is ejected if $w^+ \leq w_\alpha^*$. The parameter w_α^* (critical value of w^+ for the signed-rank test) here is btained from the standard table of critical values for a particular significance level α.

Table 5.3: Differences and Signed-ranks Using the ENGA-NSGA Test Data from Table 5.2

Seed used	Difference	Signed Rank
760	-200	-1
1997	-700	-2
1835	-1000	-3
35	-2700	-4.5
1000	-2700	-4.5
1520	-3000	-6.5
7494	-3000	-6.5
18	-3800	-8
1234	-5600	-9
1231	-8200	-10

rom the Table 5.3, the values calculated are $w^+ = 0$ and $w^- = 55$. Therefore $w = \min(0, 55)$ which gives $w = 0$. For the significance level (α) of 0.05, the critical value (w_α^*) is 10. Since $w^+ = 0$ is less than $w_{0.05}^*$ (10), the null hypothesis is rejected which suggests that with identical arameterization (Table 5. 1) the mean function evaluations for ENGA are less than that of NSGA.

5.1.3 Criteria 2 : Comparison on the Basis of Number of Distinct Nondominated (Efficient) Solutions Found

In this case comparison is based on a different metric, as follows. From the calculation of umber of function evaluations for the both algorithms for each of 10 different seeds (obtained rom criteria 1), the count of evaluations for a chosen identical seed for the two methods

NSGA and ENGA) are taken. Then identically parameterized NSGA and ENGA are xecuted for the problem for the lower of the evaluation number of the two. When the run erminates the number of nondominated solutions found and their quality are compared.

While the above test was being conducted, another fact observed was that ENGA, after completing the required function evaluations to satisfy the desired optimality target puts a arge number of solutions on Front 1. To compare the two algorithms under such conditions, Test Problem 1 was optimized using the two algorithms with the identical parameter settings Table 5.1). Table 5.4 displays the count of efficient solutions noted on the nondominated front for *each* of the ten seeds at the point where either NSGA or ENGA satisfied the pre-set arget.

Table 5.4: Number of Solutions Delivered on the Pareto Front for Test Problem 1 at the Generation at which the Target is First Satisfied.

Generation at which Target was Satisfied by ENGA / NSGA	Seed	Number on Front1 by NSGA	Number on Front 1 by ENGA
3	1000	68	98
3	1520	83	95
5	760	80	98
5	7494	83	99
5	1835	87	97
5	1234	77	97
5	1997	85	98
5	18	84	99
5	35	78	99
4	1231	85	96

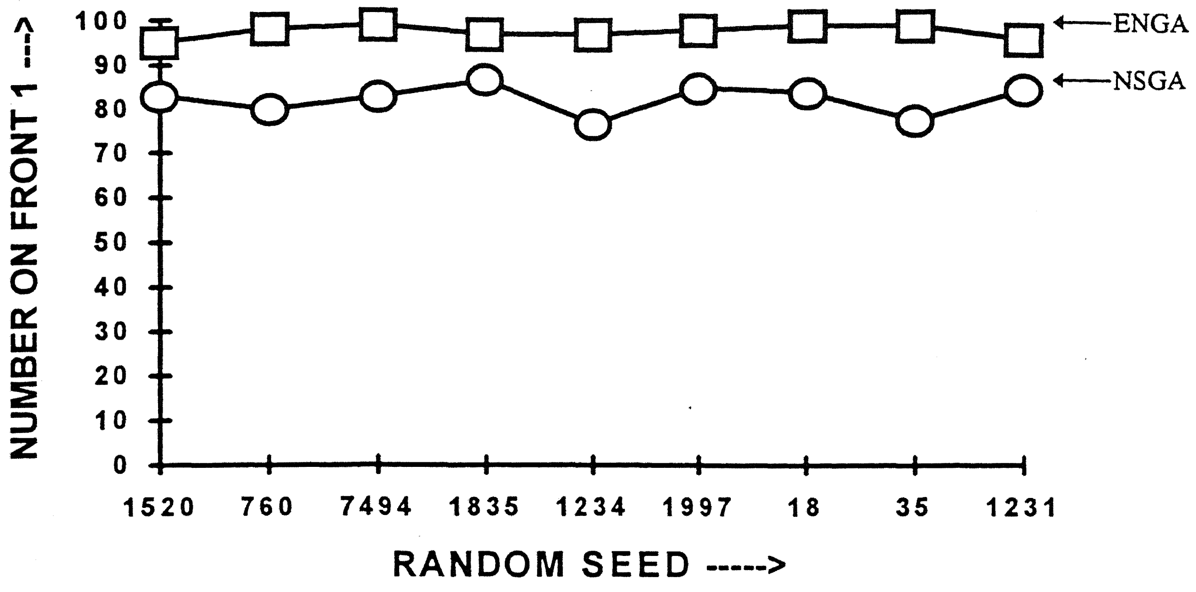

Figure 5.2: Number on Front 1 for NSGA - ENGA at the Time the Target Gets Satisfied for the Test Problem 1

As Table 5.4 shows by the time ENGA had placed almost all the members on the Pareto frontier, NSGA found only around 80 - 85 % of the efficient solutions (Figure 5.2 displays this comparison graphically). This again demonstrates the faster convergence capability of ENGA over NSGA. Additionally, the efficient solutions at the end of generation at which the comparison was made were compared qualitatively. Their location on the efficient front showed that distribution of the solutions found by ENGA along the entire nondominated front is wider than that achieved by NSGA, i.e., for the test problem ENGA found more *unique* nondominated solutions in number.

5.1.4: Statistical Test for Number of Nondominated Solutions Found

The ranked sign test was repeated, as follows, for the expected number (μ) of members on the nondominated front (Front 1) when the preset target gets satisfied.

1. The response of interest is the count of nondominated solutions produced by the two algorithms.
2. H_0: $\mu_1 = \mu_2$ or, equivalently, $H_0 = \mu_{\text{Difference}} = 0$
3. H_1: $\mu_1 > \mu_2$ or, equivalently, H_1: $\mu_{\text{Difference}} > 0$
4. $\alpha = 0.05$
5. Since $\alpha = 0.05$, the critical value $w_{0.05}^* = 10$. We reject H_0 if $w^- \leq w_{0.05}^*$

Table 5.5: Differences and Signed-ranks Using the Data from Table 5.4

Seed used	Difference	Signed Rank
1835	+10	+1
1231	+11	+2
1520	+12	+3
1997	+13	+4
18	+15	+5
7494	+16	+6
760	+18	+7
1234	+20	+8
35	+21	+9
1000	+30	+10

From the Table 5.5 we can compute $w^+ = 55$ and $w^- = 0$. Since $w^- = 0$ is less than as $w_{0.05}^* = 10$, we reject the null hypothesis. Thus we conclude that the mean number of nondominated members produced by ENGA is greater than that of NSGA.

5.2 Test Problem 2 : The Electronic Filter Design Problem

This is the two objective optimization problem described in Chapter 3 to achieve robust design of an electronic filter. The two objectives are

1) Minimize variance in cut-off frequency ω_c (= f_1 (x))

2) Minimize variance in galvanometer deflection D(= f_2 (x))

For this problem also the statistical comparison is done as done for Test Problem 1. Table 5.6 shows the optimum GA parameter values (found by DOE) used for this problem.

Table 5.6: Parameter values for Electronic Filter Problem

Parameter	Value
Population size	100
Pcross	0.8
Pmute	0.01
Length of chromosome	32
Dshare	0.1
$R_{3\,min}$ (Lower limit of R_3)	1
$R_{3\,max}$ (Upper limit of R_3)	500

Table 5.7: Number of Function Evaluations Required for Test Problem 2 to obtain 95% or More Members on the Efficient Front.

Seed used	Total Function Evaluations for NSGA	Total Function Evaluations for ENGA
760	10300	900
7494	1300	300
1835	1500	300
1520	5100	400
1000	5700	700
1234	2400	300
1997	11100	300
18	2600	300
35	800	900
1231	2100	1000

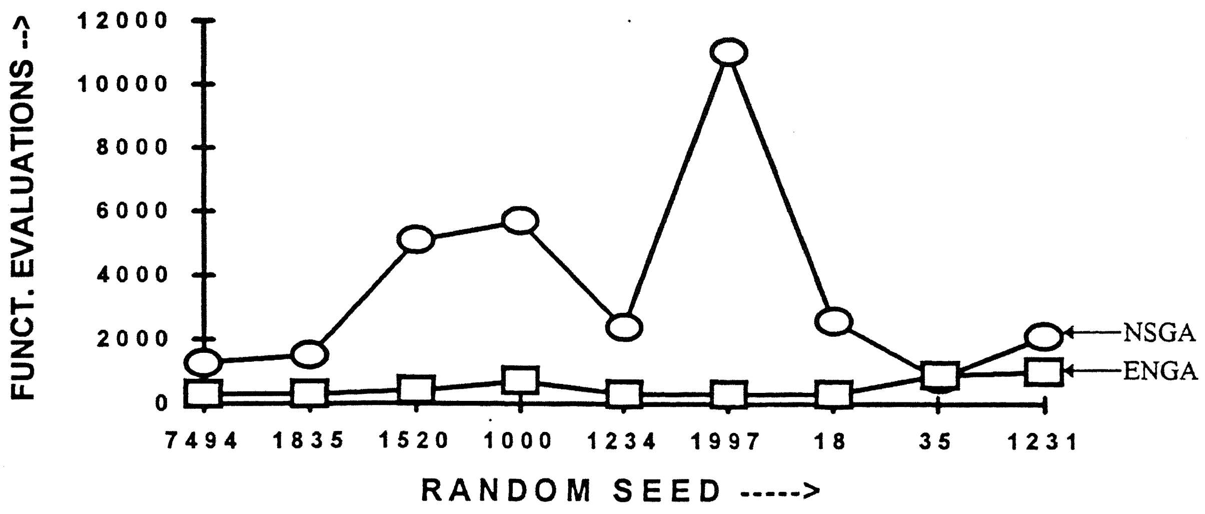

Figure 5.3: Comparison of NSGA - ENGA on Function Evaluations for Test Problem 2

As Table 5.7 shows, in this case also ENGA converges considerably faster than NSGA (Figure 5.3 shows the typical number of function evaluations required for both ENGA and NSGA). In this application ENGA was not as consistent in finding 95% of the efficient solutions for the different starting conditions used as it was in Problem 1. Still it required considerably fewer evaluations when compared with NSGA.

5.2.1 Statistical Test for Number of Function Evaluations for the Filter Problem

The test proceeded along the following steps

1. The parameters of interest are the number of function evaluations for the two algorithms.

2. Set H_0: $\mu_1 = \mu_2$ or, equivalently, $H_0 = \mu_{Difference} = 0$

3. Set H_1: $\mu_1 < \mu_2$ or, equivalently, H_1: $\mu_{Difference} < 0$

4. Set $\alpha = 0.05$

5. Since $\alpha = 0.05$, the critical value is $w_{0.05}^{*} = 10$. Hence we would reject H_0 if $w^{+} \leq w_{0.05}^{*}$

Table 5.8: Differences and Signed-ranks Using the Data from Table 5.7

Seed used	Difference	Signed Rank
35	+100	1
7494	-1000	-2
1231	-1100	-3
1835	-1200	-4
1234	-2100	-5
18	-2300	-6
1520	-4700	-7
1000	-5000	-8
760	-9400	-9
1997	-10800	-10

From the Table 5.8, we can compute $w^{+} = 1$ and $w^{-} = 54$. Since $w^{+} = 1$ is less than as $w_{0.05}^{*} = 10$, we reject the null hypothesis. Thus we accept that the mean number of function evaluations required to satisfy the target by ENGA is less than that of NSGA.

To compare the number of nondominated solutions produced, both NSGA and ENGA were run upto generation at which either of them satisfied the set target. Table 5.9 shows the count of Front 1 (i.e. efficient members for different seeds at this point

From Table 5.9 one may observe that starting from a set of identical random initial solutions ENGA was able to put almost entire population on the Pareto front while NSGA put about 80% of the population on that front Figure 5.4 shows the results graphically.

Table 5.9: Number of Solutions on the Pareto Front for Test Problem 2 at the Generation at which Target is First Satisfied

Generation at which Target was Satisfied by ENGA / NSGA	Seed	Number on Front1 by NSGA	Number on Front 1 by ENGA
6	1000	81	98
3	1520	78	96
8	760	87	97
2	7494	74	97
2	1835	70	98
2	1234	78	96
2	1997	71	98
2	18	71	97
7	35	96	88
9	1231	92	98

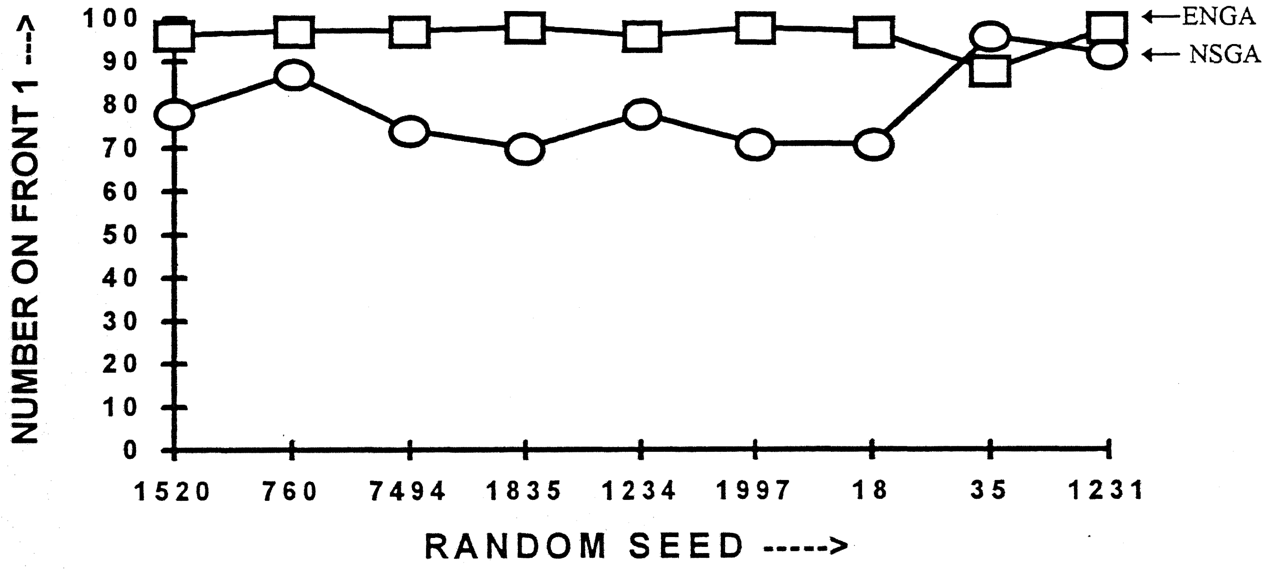

Figure 5.4: Membership Count on Front 1 for NSGA - ENGA at the Instant When the Target Gets Satisfied for Test Problem 2

5.2.2 Statistical Test for The Number of Nondominated Solutions for the Filter Design Problem

The steps were as follows:

1. The Response of interest is the number of nondominated solutions produced by the two algorithms.
2. H_0: $\mu_1 = \mu_2$ or, equivalently, $H_0 = \mu_{Difference} = 0$
3. H_1: $\mu_1 > \mu_2$ or, equivalently, H_1: $\mu_{Difference} > 0$
4. $\alpha = 0.05$
5. Since $\alpha = 0.05$, the critical value $w_{0.05}{}^{*} = 10$. We would reject H_0 if $w^{-} \leq w_{0.05}{}^{*}$

Table 5.10: Differences and Signed-ranks Using the Data from Table 5.9

Seed used	Difference	Signed Rank
1231	+6	+1
35	-8	-2
760	+10	+3
1000	+17	+4
1520	+18	+5.5
1234	+18	+5.5
7494	+23	+7
18	+26	+8
1997	+27	+9
1835	+28	+10

From the Table 5.10 we can compute $w^{+} = 53$ and $w^{-} = 2$. Since $w^{-} = 2$ is less than as $w_{0.05}{}^{*} = 10$, we reject the null hypothèsis. Thus we accept that the mean number of nondominated members of ENGA is greater than that of NSGA.

For the filter design problem, Figures 5.5 to 5.7 (a display of the results in Tables 5.11 to 5.13) show the initial population and populations at 5th and 10th generations respectively for

ENGA. Figures 5.8 to 5.10 (Tables 5.14 to 5.16) show the same data for NSGA. One may observe that in the two cases the initial distribution on the efficient front is exactly alike, as expected. However, at the 10th generation we see the difference developing along the Pareto front between ENGA and NSGA. In NSGA clustering of solutions occurs because of the duplication of many solutions. For ENGA we do not observe much clustering.

5.3 Conclusions

The algorithms NSGA and ENGA were compared in this chapter using two distinct multiobjective optimization problems with respect to two evaluation criteria (a) the number of function evaluations required to get 95% of the population on the nondominated front and (b) the number of nondominated solutions at the point the above target gets satisfied. The comparison used the Wilcoxon signed-rank test. In both the cases ENGA showed superior capability in finding the efficient solutions for comparable computational effort.

Table 5.11: Initial Population Created by ENGA for the Filter Problem

R_3	R_2	C	Variance - ω_c	Variance D
273.94815063	45.53411865	440.77120972	0.14199065	0.01791839
6.18465042	604.75286865	3987.6677246	0.25556216	0.01894961
102.50369263	248.59353638	487.83929443	0.16656049	0.01733351
237.29531860	72.56854248	428.20196533	0.14496864	0.01757595
155.21667480	159.22561646	434.09432983	0.15422811	0.01733576
255.76417542	58.28706360	433.58471680	0.14343379	0.01772438
334.99969482	10.17328644	478.26907349	0.13774367	0.01898192
299.57006836	29.44085693	453.97256470	0.14005722	0.01828241
324.27429199	15.68170929	470.14691162	0.13837306	0.01874465
345.50265503	5.02510834	486.90280151	0.13722166	0.01923678
91.30902100	273.31259155	511.19125366	0.17066289	0.01733639
328.97061157	13.23735046	473.61883545	0.13808735	0.01884564
187.48757935	119.63568115	424.75076294	0.14983504	0.01736981
314.13037109	21.14193726	463.08651733	0.13903339	0.01854090
63.81732559	346.79324341	610.57635498	0.18469922	0.01738186
184.00854492	123.49952698	425.23196411	0.15024321	0.01736349
56.46617508	370.32113647	655.39721680	0.18977644	0.01741747
56.19887161	371.21365356	657.26190186	0.18997477	0.01741911
194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
298.59381104	30.01794434	453.40371704	0.14012796	0.01826641
136.41842651	186.88037109	445.96548462	0.15764675	0.01733219
127.72585297	201.08734131	453.69073486	0.15953584	0.01733207
157.60852051	155.97546387	432.97744751	0.15384702	0.01733676
130.71774292	196.08630371	450.84515381	0.15886033	0.01733210
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
261.12792969	54.39875793	435.51388550	0.14300199	0.01777642
9.37910938	584.43853760	2707.7690429	0.24877964	0.01870634
50.81306458	389.79177856	699.24102783	0.19419040	0.01745811
286.56909180	37.35334778	446.82592773	0.14102177	0.01808375
60.68158722	356.59527588	628.28240967	0.18678030	0.01739508
239.27679443	70.96601868	428.68356323	0.14479959	0.01758970
16.58937263	542.07952881	1632.1982421	0.23529379	0.01827629
160.99842834	151.46392822	431.52639771	0.15332527	0.01733855

320.47332764	17.69821167	467.43173218	0.13861413	0.01866598
187.42239380	119.70727539	424.75872803	0.14984219	0.01736974
205.20266724	101.24331665	423.90643311	0.14792614	0.01741602
320.47332764	17.69821167	467.43173218	0.13861413	0.01866598
136.41842651	186.88037109	445.96548462	0.15764675	0.01733219
194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
187.56791687	119.54747009	424.74096680	0.14982560	0.01737004
279.04177856	42.17019653	443.11026001	0.14159635	0.01798192
237.29531860	72.56854248	428.20196533	0.14496864	0.01757595
336.76651001	9.29069519	479.67358398	0.13764845	0.01902316
96.03188324	262.57153320	500.52972412	0.16884191	0.01733480
161.65542603	150.60206604	431.26251221	0.15322593	0.01733890
97.32568359	259.71060181	497.82583618	0.16836691	0.01733446
288.07217407	36.41278076	447.60491943	0.14090855	0.01810520
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
123.62770844	208.13699341	457.93640137	0.16050823	0.01733216
278.90325928	42.26054382	443.04483032	0.14160709	0.01798013
324.01022339	15.82068634	469.95559692	0.13838944	0.01873903
40.59722900	428.45434570	810.92303467	0.20352469	0.01757443
127.78094482	200.99420166	453.63638306	0.15952329	0.01733207
334.94451904	10.20096588	478.22552490	0.13774672	0.01898062
298.38717651	30.14044189	453.28399658	0.14014265	0.01826313
26.61920547	490.08447266	1107.1384277	0.21995679	0.01788006
187.57009888	119.54510498	424.74072266	0.14982513	0.01737005
286.56954956	37.35305786	446.82620239	0.14102158	0.01808375
299.57006836	29.44085693	453.97256470	0.14005722	0.01828241
2.00376129	632.94738770	11833.267578	0.26531175	0.01932894
205.19830322	101.24760437	423.90634155	0.14792642	0.01741602
265.42239380	51.36363220	437.17398071	0.14266017	0.01782114
334.51629639	10.41596222	477.88809204	0.13777028	0.01897073
273.89901733	45.56697083	440.74935913	0.14199446	0.01791779
253.37292480	60.05653381	432.77694702	0.14362822	0.01770253
123.48920441	208.37939453	458.08721924	0.16054198	0.01733221
331.12860107	12.13121796	475.25817871	0.13796124	0.01889359
55.11863708	374.84826660	664.99206543	0.19078597	0.01742592
331.46942139	11.95749664	475.51956177	0.13794120	0.01890127
301.87765503	28.08740234	455.33819580	0.13988979	0.01832076

324.01062012	15.82048798	469.95587158	0.13838972	0.01873907
265.42239380	51.36363220	437.17398071	0.14266017	0.01782114
34.01773834	456.07025146	919.53118896	0.21065250	0.01769139
128.49368286	199.79241943	452.94064331	0.15935948	0.01733212
160.99842834	151.46392822	431.52639771	0.15332527	0.01733855
128.49368286	199.79241943	452.94064331	0.15935948	0.01733212
300.45080566	28.92253876	454.49026489	0.13999304	0.01829698
337.78219604	8.78640747	480.48965454	0.13759467	0.01904722
95.99696350	262.64926147	500.60388184	0.16885491	0.01733478
187.42677307	119.70246887	424.75817871	0.14984180	0.01736975
187.48751831	119.63574219	424.75073242	0.14983453	0.01736984
324.27444458	15.68164062	470.14703369	0.13837311	0.01874462
216.40852356	90.60923767	424.61898804	0.14683661	0.01745942
334.51629639	10.41596222	477.88809204	0.13777028	0.01897073
161.65547180	150.60203552	431.26242065	0.15322649	0.01733894
176.21383667	132.48591614	426.72918701	0.15120767	0.01735207
312.23025513	22.19322205	461.82962036	0.13916263	0.01850495
109.63808441	234.09738159	476.00317383	0.16429837	0.01733275
337.60675049	8.87335968	480.34826660	0.13760400	0.01904306
51.84040451	386.15814209	690.52777100	0.19335234	0.01744969
216.43792725	90.58226013	424.62203979	0.14683357	0.01745955
20.43684196	521.25604248	1369.4705810	0.22899042	0.01810098
337.78213501	8.78643799	480.48956299	0.13759497	0.01904722
92.10993195	271.45742798	509.29083252	0.17034395	0.01733611
101.98915100	249.67498779	488.77542114	0.16673329	0.01733359
271.05447388	47.48417664	439.50561523	0.14221649	0.01788411
236.72409058	73.03382874	428.06750488	0.14501779	0.01757213
130.71815491	196.08560181	450.84484863	0.15885974	0.01733202
282.50054932	39.93450928	444.77923584	0.14133087	0.01802753
1.71385002	634.97418213	13796.700195	0.26602831	0.01935815

Table 5.12 : Population at the Generation 5 by ENGA for the Filter Problem

R_3	R_2	C	Variance - ω_c	Variance D
237.29531860	72.56854248	428.20196533	0.14496864	0.01757595
188.91516113	118.07536316	424.58502197	0.14967081	0.01737268
305.55755615	25.95942688	457.57690430	0.13962725	0.01838399
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
283.83197021	39.08410645	445.43908691	0.14122929	0.01804560
299.57006836	29.44085693	453.97256470	0.14005722	0.01828241
273.89898682	45.56700134	440.74935913	0.14199437	0.01791778
239.31546021	70.93489075	428.69323730	0.14479634	0.01759004
187.57054138	119.54460144	424.74060059	0.14982505	0.01737000
187.48779297	119.63545227	424.75073242	0.14983475	0.01736986
136.03402710	187.48818970	446.27343750	0.15772568	0.01733212
338.12948608	8.61447906	480.77014160	0.13757679	0.01905547
216.40852356	90.60923767	424.61898804	0.14683661	0.01745942
324.27423096	15.68174744	470.14691162	0.13837306	0.01874464
345.50265503	5.02510834	486.90280151	0.13722166	0.01923678
298.59381104	30.01794434	453.40371704	0.14012796	0.01826641
320.52209473	17.67211914	467.46606445	0.13861069	0.01866691
336.41885376	9.46382141	479.39566040	0.13766694	0.01901503
299.57006836	29.44085693	453.97256470	0.14005722	0.01828241
240.54821777	69.94699097	429.00494385	0.14469199	0.01759880
187.93542480	119.14466858	424.69677734	0.14978281	0.01737071
252.86425781	60.43585205	432.60925293	0.14366969	0.01769801
285.34561157	38.12409973	446.20098877	0.14111450	0.01806652
298.61511230	30.00532532	453.41604614	0.14012632	0.01826679
194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
188.91516113	118.07536316	424.58502197	0.14967081	0.01737268
183.49256897	124.08015442	425.31286621	0.15030532	0.01736262
160.89164734	151.60435486	431.56982422	0.15334141	0.01733846
306.14401245	25.62369537	457.94076538	0.13958554	0.01839423
135.13398743	188.91867065	447.00598145	0.15791206	0.01733211
324.27429199	15.68170929	470.14691162	0.13837306	0.01874465
136.03395081	187.48834229	446.27340698	0.15772523	0.01733216
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113

194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
314.13076782	21.14171600	463.08682251	0.13903305	0.01854091
216.40805054	90.60966492	424.61892700	0.14683677	0.01745941
314.13024902	21.14200592	463.08648682	0.13903332	0.01854089
160.99844360	151.46389771	431.52639771	0.15332521	0.01733854
299.57006836	29.44085693	453.97256470	0.14005722	0.01828241
205.19830322	101.24760437	423.90634155	0.14792642	0.01741602
331.46942139	11.95749664	475.51956177	0.13794120	0.01890127
155.52050781	158.80963135	433.94808960	0.15417953	0.01733586
324.01062012	15.82048798	469.95587158	0.13838972	0.01873907
324.27429199	15.68170929	470.14691162	0.13837306	0.01874465
22.33757210	511.38421631	1273.2377929	0.22607681	0.01802580
309.63400269	23.64469910	460.14523315	0.13934128	0.01845684
271.01641846	47.51000977	439.48919678	0.14221944	0.01788362
331.48907471	11.94748688	475.53472900	0.13794030	0.01890170
225.81921387	82.21112061	425.88690186	0.14597258	0.01750579
288.07098389	36.41352844	447.60430908	0.14090846	0.01810523
165.21922302	145.99642944	429.92501831	0.15270264	0.01734127
265.42239380	51.36363220	437.17398071	0.14266017	0.01782114
161.65547180	150.60203552	431.26242065	0.15322649	0.01733894
187.57055664	119.54458618	424.74060059	0.14982519	0.01737000
324.27807617	15.67972565	470.14968872	0.13837276	0.01874467
346.64062500	4.48122406	487.87997437	0.13717203	0.01926575
1.71395898	634.97351074	13795.837890	0.26602766	0.01935812
187.48753357	119.63569641	424.75079346	0.14983477	0.01736982
345.50253296	5.02516174	486.90270996	0.13722154	0.01923673
165.21934509	145.99627686	429.92495728	0.15270258	0.01734134
160.80838013	151.71395874	431.60379028	0.15335417	0.01733839
210.05708313	96.54797363	424.10440063	0.14744486	0.01743333
160.99842834	151.46392822	431.52639771	0.15332527	0.01733855
176.21383667	132.48591614	426.72918701	0.15120767	0.01735207
271.03616333	47.49662781	439.49771118	0.14221765	0.01788390
279.04168701	42.17025757	443.11026001	0.14159617	0.01798197
237.29809570	72.56626892	428.20260620	0.14496855	0.01757597
273.89089966	45.57241821	440.74575806	0.14199492	0.01791768
274.67675781	45.04760742	441.09710693	0.14193402	0.01792717
250.87947083	61.92593384	431.96914673	0.14383198	0.01768060

215.33967590	91.59289551	424.51266479	0.14693740	0.01745476
253.37287903	60.05656433	432.77688599	0.14362839	0.01770254
347.15060425	4.23834991	488.32058716	0.13715032	0.01927882
208.95300293	97.60366821	424.04418945	0.14755288	0.01742924
225.81767273	82.21246338	425.88659668	0.14597321	0.01750573
196.83757019	109.67190552	423.98550415	0.14879480	0.01739111
337.77743530	8.78876495	480.48574829	0.13759525	0.01904715
283.83666992	39.08110046	445.44143677	0.14122911	0.01804567
299.58322144	29.43309784	453.98031616	0.14005588	0.01828257
306.13085938	25.63121033	457.93255615	0.13958682	0.01839402
205.19827271	101.24763489	423.90631104	0.14792641	0.01741602
321.71496582	17.03567505	468.30944824	0.13853431	0.01869132
136.03004456	187.49450684	446.27661133	0.15772657	0.01733213
161.65921021	150.59713745	431.26095581	0.15322585	0.01733893
205.19827271	101.24763489	423.90631104	0.14792641	0.01741602
194.77735901	111.81652832	424.09048462	0.14901707	0.01738585
271.03607178	47.49667358	439.49771118	0.14221781	0.01788392
228.40919495	79.97943115	426.33724976	0.14574170	0.01752022
165.21916199	145.99650574	429.92501831	0.15270264	0.01734127
196.83763123	109.67182922	423.98547363	0.14879479	0.01739118
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
271.03607178	47.49667358	439.49771118	0.14221781	0.01788392
324.01062012	15.82048798	469.95587158	0.13838972	0.01873907
285.34695435	38.12326050	446.20166016	0.14111425	0.01806653
305.55743408	25.95949554	457.57690430	0.13962702	0.01838392
136.03405762	187.48815918	446.27340698	0.15772571	0.01733213
194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
187.48767090	119.63555908	424.75073242	0.14983520	0.01736986
285.34558105	38.12411499	446.20098877	0.14111461	0.01806651
165.21926880	145.99638367	429.92498779	0.15270257	0.01734127

Table 5.13 : Population obtained at the Generation 10 by ENGA for Filter Problem

R_3	R_2	C	Variance - ω_c	Variance D
285.34561157	38.12409973	446.20098877	0.14111450	0.01806652
155.52050781	158.80963135	433.94808960	0.15417953	0.01733586
163.62945557	148.03666687	430.50225830	0.15293372	0.01734016
314.13027954	21.14198303	463.08648682	0.13903324	0.01854089
306.14401245	25.62369537	457.94076538	0.13958554	0.01839423
279.04177856	42.17019653	443.11026001	0.14159635	0.01798192
187.14685059	120.01025391	424.79296875	0.14987423	0.01736922
187.57058716	119.54455566	424.74057007	0.14982519	0.01737000
205.19738770	101.24850464	423.90631104	0.14792670	0.01741603
187.57150269	119.54354858	424.74044800	0.14982507	0.01737001
225.81767273	82.21246338	425.88659668	0.14597321	0.01750573
225.81800842	82.21218872	425.88662720	0.14597283	0.01750574
273.87036133	45.58616638	440.73660278	0.14199659	0.01791738
155.54914856	158.77047729	433.93432617	0.15417454	0.01733587
273.93038940	45.54598999	440.76333618	0.14199209	0.01791818
345.47128296	5.04013062	486.87603760	0.13722320	0.01923595
191.47009277	115.31861877	424.33325195	0.14938198	0.01737815
240.46257019	70.01542664	428.98303223	0.14469944	0.01759821
187.12008667	120.03970337	424.79635620	0.14987710	0.01736913
176.66430664	131.95372009	426.62615967	0.15114956	0.01735264
263.14849854	52.96223450	436.28219604	0.14284098	0.01779713
300.86239624	28.68106079	454.73373413	0.13996324	0.01830373
225.81767273	82.21246338	425.88659668	0.14597321	0.01750573
285.34561157	38.12409973	446.20098877	0.14111450	0.01806652
306.14401245	25.62369537	457.94076538	0.13958554	0.01839423
209.45652771	97.12130737	424.07055664	0.14750347	0.01743108
331.46975708	11.95732880	475.51986694	0.13794164	0.01890126
165.21922302	145.99642944	429.92501831	0.15270264	0.01734127
195.14129639	111.43565369	424.06936646	0.14897771	0.01738680
331.10546875	12.14302826	475.24047852	0.13796255	0.01889309
225.81767273	82.21246338	425.88659668	0.14597321	0.01750573
187.12008667	120.03970337	424.79635620	0.14987710	0.01736913
216.40852356	90.60923767	424.61898804	0.14683661	0.01745942

197.17839050	109.31983948	423.97137451	0.14875853	0.01739203
160.99636841	151.46662903	431.52725220	0.15332530	0.01733850
345.50469971	5.02411652	486.90457153	0.13722162	0.01923683
265.42239380	51.36363220	437.17398071	0.14266017	0.01782114
205.19830322	101.24760437	423.90634155	0.14792642	0.01741602
309.65692139	23.63181305	460.15994263	0.13933946	0.01845731
331.46951294	11.95745087	475.51965332	0.13794123	0.01890128
324.27429199	15.68170929	470.14691162	0.13837306	0.01874465
250.53211975	62.18833923	431.85946655	0.14386044	0.01767763
273.90948486	45.55998230	440.75399780	0.14199328	0.01791787
314.13046265	21.14189148	463.08657837	0.13903317	0.01854089
174.76948547	134.20327759	427.07366943	0.15139475	0.01735030
270.71795654	47.71284485	439.36135864	0.14224277	0.01788017
262.99276733	53.07241821	436.22219849	0.14285304	0.01779550
136.03395081	187.48834229	446.27340698	0.15772523	0.01733216
314.13027954	21.14198303	463.08648682	0.13903324	0.01854089
337.78213501	8.78643799	480.48956299	0.13759497	0.01904722
165.13371277	146.10557556	429.95532227	0.15271516	0.01734122
216.40884399	90.60894775	424.61907959	0.14683637	0.01745937
176.21383667	132.48591614	426.72918701	0.15120767	0.01735207
240.54821777	69.94699097	429.00494385	0.14469199	0.01759880
213.60426331	93.20339966	424.35687256	0.14710189	0.01744741
274.90200806	44.89756775	441.19848633	0.14191660	0.01792991
154.70802307	159.92407227	434.34216309	0.15431058	0.01733553
188.23925781	118.81231689	424.66119385	0.14974794	0.01737132
187.40246582	119.72917175	424.76119995	0.14984433	0.01736970
187.51196289	119.60891724	424.74777222	0.14983205	0.01736988
346.86593628	4.37384796	488.07440186	0.13716277	0.01927150
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
286.89978027	37.14581299	446.99630737	0.14099696	0.01808845
233.54092407	75.65420532	427.35464478	0.14529219	0.01755126
205.19830322	101.24760437	423.90634155	0.14792642	0.01741602
237.29531860	72.56854248	428.20196533	0.14496864	0.01757595
258.96463013	55.95373535	434.71640015	0.14317532	0.01775491
278.87136841	42.28134155	443.02966309	0.14160934	0.01797977
194.77394104	111.82011414	424.09072876	0.14901733	0.01738582
279.04516602	42.16798401	443.11187744	0.14159594	0.01798202

187.48782349	119.63543701	424.75070190	0.14983495	0.01736985
196.83757019	109.67190552	423.98550415	0.14879480	0.01739111
188.07400513	118.99298096	424.68038940	0.14976706	0.01737104
214.83621216	92.05839539	424.46530151	0.14698491	0.01745257
285.34561157	38.12409973	446.20098877	0.14111450	0.01806652
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
314.13027954	21.14198303	463.08648682	0.13903324	0.01854089
283.83197021	39.08410645	445.43908691	0.14122929	0.01804560
240.54817200	69.94705200	429.00494385	0.14469220	0.01759882
165.21925354	145.99639893	429.92495728	0.15270278	0.01734127
187.48767090	119.63555908	424.75073242	0.14983520	0.01736986
161.65547180	150.60203552	431.26242065	0.15322649	0.01733894
298.59381104	30.01794434	453.40371704	0.14012796	0.01826641
215.33963013	91.59292603	424.51266479	0.14693747	0.01745473
326.56542969	14.48279572	471.82440186	0.13823174	0.01879340
258.25265503	56.46940613	434.45971680	0.14323303	0.01774797
165.21922302	145.99642944	429.92501831	0.15270264	0.01734127
298.59381104	30.01794434	453.40371704	0.14012796	0.01826641
291.78823853	34.11691284	449.58370972	0.14062984	0.01815991
175.27967834	133.59474182	426.94946289	0.15132813	0.01735086
298.59387207	30.01791382	453.40377808	0.14012815	0.01826641
337.78207397	8.78646851	480.48950195	0.13759489	0.01904722
134.38961792	190.10931396	447.62432861	0.15806806	0.01733210
337.73956299	8.80752563	480.45523071	0.13759734	0.01904616
237.02429199	72.78910828	428.13793945	0.14499155	0.01757411
326.56542969	14.48279572	471.82440186	0.13823174	0.01879340
176.21383667	132.48591614	426.72918701	0.15120767	0.01735207
205.15570068	101.28942871	423.90536499	0.14793144	0.01741588
218.89401245	88.34585571	424.89648438	0.14660391	0.01747072
302.58963013	27.67280579	455.76547241	0.13983877	0.01833280

Table 5.14 : Initial Population Created by NSGA for the Filter Problem

R_3	R_2	C	Variance - ω_c	Variance D
273.94815063	45.53411865	440.77120972	0.14199065	0.01791839
6.18465042	604.75286865	3987.66772461	0.25556216	0.01894961
102.50369263	248.59353638	487.83929443	0.16656049	0.01733351
237.29531860	72.56854248	428.20196533	0.14496864	0.01757595
155.21667480	159.22561646	434.09432983	0.15422811	0.01733576
255.76417542	58.28706360	433.58471680	0.14343379	0.01772438
334.99969482	10.17328644	478.26907349	0.13774367	0.01898192
299.57006836	29.44085693	453.97256470	0.14005722	0.01828241
324.27429199	15.68170929	470.14691162	0.13837306	0.01874465
345.50265503	5.02510834	486.90280151	0.13722166	0.01923678
91.30902100	273.31259155	511.19125366	0.17066289	0.01733639
328.97061157	13.23735046	473.61883545	0.13808735	0.01884564
187.48757935	119.63568115	424.75076294	0.14983504	0.01736981
314.13037109	21.14193726	463.08651733	0.13903339	0.01854090
63.81732559	346.79324341	610.57635498	0.18469922	0.01738186
184.00854492	123.49952698	425.23196411	0.15024321	0.01736349
56.46617508	370.32113647	655.39721680	0.18977644	0.01741747
56.19887161	371.21365356	657.26190186	0.18997477	0.01741911
194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
298.59381104	30.01794434	453.40371704	0.14012796	0.01826641
136.41842651	186.88037109	445.96548462	0.15764675	0.01733219
127.72585297	201.08734131	453.69073486	0.15953584	0.01733207
157.60852051	155.97546387	432.97744751	0.15384702	0.01733676
130.71774292	196.08630371	450.84515381	0.15886033	0.01733210
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
261.12792969	54.39875793	435.51388550	0.14300199	0.01777642
9.37910938	584.43853760	2707.76904297	0.24877964	0.01870634
50.81306458	389.79177856	699.24102783	0.19419040	0.01745811
286.56909180	37.35334778	446.82592773	0.14102177	0.01808375
60.68158722	356.59527588	628.28240967	0.18678030	0.01739508
239.27679443	70.96601868	428.68356323	0.14479959	0.01758970
16.58937263	542.07952881	1632.19824219	0.23529379	0.01827629
160.99842834	151.46392822	431.52639771	0.15332527	0.01733855

324.01062012	15.82048798	469.95587158	0.13838972	0.01873907
265.42239380	51.36363220	437.17398071	0.14266017	0.01782114
34.01773834	456.07025146	919.53118896	0.21065250	0.01769139
128.49368286	199.79241943	452.94064331	0.15935948	0.01733212
160.99842834	151.46392822	431.52639771	0.15332527	0.01733855
128.49368286	199.79241943	452.94064331	0.15935948	0.01733212
300.45080566	28.92253876	454.49026489	0.13999304	0.01829698
337.78219604	8.78640747	480.48965454	0.13759467	0.01904722
95.99696350	262.64926147	500.60388184	0.16885491	0.01733478
187.42677307	119.70246887	424.75817871	0.14984180	0.01736975
187.48751831	119.63574219	424.75073242	0.14983453	0.01736984
324.27444458	15.68164062	470.14703369	0.13837311	0.01874462
216.40852356	90.60923767	424.61898804	0.14683661	0.01745942
334.51629639	10.41596222	477.88809204	0.13777028	0.01897073
161.65547180	150.60203552	431.26242065	0.15322649	0.01733894
176.21383667	132.48591614	426.72918701	0.15120767	0.01735207
312.23025513	22.19322205	461.82962036	0.13916263	0.01850495
109.63808441	234.09738159	476.00317383	0.16429837	0.01733275
337.60675049	8.87335968	480.34826660	0.13760400	0.01904306
51.84040451	386.15814209	690.52777100	0.19335234	0.01744969
216.43792725	90.58226013	424.62203979	0.14683357	0.01745955
20.43684196	521.25604248	1369.47058105	0.22899042	0.01810098
337.78213501	8.78643799	480.48956299	0.13759497	0.01904722
92.10993195	271.45742798	509.29083252	0.17034395	0.01733611
101.98915100	249.67498779	488.77542114	0.16673329	0.01733359
271.05447388	47.48417664	439.50561523	0.14221649	0.01788411
236.72409058	73.03382874	428.06750488	0.14501779	0.01757213
130.71815491	196.08560181	450.84484863	0.15885974	0.01733202
282.50054932	39.93450928	444.77923584	0.14133087	0.01802753
1.71385002	634.97418213	13796.7001953	0.26602831	0.01935815

Table 5.15: Population at Generation 5 by NSGA for the Filter Problem

R_3	R_2	C	Variance - ω_c	Variance D
271.82583618	46.96154785	439.83853149	0.14215598	0.01789305
255.79142761	58.26701355	433.59417725	0.14343165	0.01772460
143.94519043	175.33456421	440.49908447	0.15617970	0.01733289
237.30201721	72.56309509	428.20352173	0.14496814	0.01757600
324.27410889	15.68180847	470.14682007	0.13837291	0.01874465
163.68769836	147.96150208	430.48059082	0.15292513	0.01734017
56.15588379	371.35742188	657.56359863	0.19000663	0.01741939
328.96862793	13.23838043	473.61734009	0.13808759	0.01884562
136.43713379	186.85079956	445.95056152	0.15764268	0.01733218
194.60076904	112.00167847	424.10113525	0.14903612	0.01738548
278.56170654	42.48356628	442.88375854	0.14163344	0.01797579
130.72018433	196.08224487	450.84298706	0.15885960	0.01733204
255.78608704	58.27096558	433.59231567	0.14343211	0.01772458
293.14071655	33.29156494	450.32260132	0.14052935	0.01818048
299.57586670	29.43743134	453.97598267	0.14005621	0.01828246
324.27410889	15.68180847	470.14682007	0.13837291	0.01874465
277.62326050	43.09835815	442.44467163	0.14170529	0.01796382
235.94984436	73.66688538	427.88845825	0.14508399	0.01756687
187.56791687	119.54747009	424.74096680	0.14982560	0.01737004
336.89932251	9.22462463	479.77993774	0.13764128	0.01902631
187.68176270	119.42257690	424.72714233	0.14981221	0.01737026
187.59140015	119.52171326	424.73809814	0.14982258	0.01737009
328.35342407	13.55566406	473.15509033	0.13812390	0.01883214
324.29541016	15.67060852	470.16226196	0.13837169	0.01874510
278.56243896	42.48309326	442.88412476	0.14163274	0.01797576
255.79142761	58.26701355	433.59417725	0.14343165	0.01772460
259.14978027	55.81994629	434.78363037	0.14316097	0.01775673
278.89318848	42.26710510	443.04003906	0.14160773	0.01798005
334.80230713	10.27233124	478.11331177	0.13775422	0.01897731
328.97061157	13.23735046	473.61883545	0.13808735	0.01884564
293.14074707	33.29154968	450.32266235	0.14052913	0.01818047
145.76509094	172.64114380	439.32852173	0.15584569	0.01733313
237.38145447	72.49848938	428.22247314	0.14496116	0.01757657

320.26513672	17.80967712	467.28546143	0.13862748	0.01866170
194.77729797	111.81660461	424.09048462	0.14901717	0.01738588
209.41664124	97.15950012	424.06835938	0.14750752	0.01743091
271.40292358	47.24783325	439.65551758	0.14218958	0.01788820
293.26031494	33.21885681	450.38845825	0.14052041	0.01818226
187.56791687	119.54747009	424.74096680	0.14982560	0.01737004
239.43214417	70.84111023	428.72232056	0.14478679	0.01759081
145.76509094	172.64114380	439.32852173	0.15584569	0.01733313
187.56785583	119.54754639	424.74096680	0.14982566	0.01737002
194.77734375	111.81652832	424.09054565	0.14901718	0.01738586
136.43707275	186.85089111	445.95062256	0.15764301	0.01733221
335.01989746	10.16316223	478.28500366	0.13774221	0.01898242
154.47061157	160.25097656	434.45910645	0.15434924	0.01733546
342.20541382	6.61609650	484.11810303	0.13737366	0.01915426
145.77572632	172.62548828	439.32183838	0.15584370	0.01733314
45.97259521	407.51760864	745.76403809	0.19837615	0.01750507
331.69366455	11.84333038	475.69204712	0.13792850	0.01890631
291.91275024	34.04071045	449.65133667	0.14062098	0.01816181
209.45915222	97.11880493	424.07067871	0.14750324	0.01743113
271.40292358	47.24783325	439.65551758	0.14218958	0.01788820
334.91970825	10.21341705	478.20596313	0.13774812	0.01897999
342.35980225	6.54109955	484.24694824	0.13736591	0.01915808
334.93969727	10.20338440	478.22167969	0.13774669	0.01898049
39.10398102	434.52154541	832.29742432	0.20505764	0.01759752
136.00906372	187.52774048	446.29348755	0.15773056	0.01733218
209.39726257	97.17800903	424.06735229	0.14750905	0.01743088
324.09259033	15.77732849	470.01525879	0.13838419	0.01874077
303.24948120	27.28979492	456.16397095	0.13979127	0.01834404
91.38043213	273.14660645	511.02020264	0.17063439	0.01733635
278.91735840	42.25134277	443.05145264	0.14160593	0.01798033
274.89669800	44.90110779	441.19601440	0.14191712	0.01792994
160.96138000	151.51261902	431.54141235	0.15333061	0.01733852
271.25885010	47.34550476	439.59344482	0.14220057	0.01788644
7.18871260	598.25781250	3462.7053222	0.25337166	0.01886917
226.11314392	81.95619202	425.93588257	0.14594629	0.01750737
100.34140015	253.17236328	491.85403442	0.16729669	0.01733382
217.97019958	89.18322754	424.78842163	0.14668989	0.01746647

298.59381104	30.01794434	453.40371704	0.14012796	0.01826641
187.68180847	119.42251587	424.72714233	0.14981207	0.01737025
334.77661133	10.28521729	478.09307861	0.13775587	0.01897671
237.47233582	72.42463684	428.24404907	0.14495358	0.01757717
209.45919800	97.11877441	424.07070923	0.14750300	0.01743113
187.56785583	119.54754639	424.74096680	0.14982566	0.01737002
325.24331665	15.17313385	470.85266113	0.13831292	0.01876511
324.27563477	15.68101501	470.14791870	0.13837300	0.01874459
345.59759521	4.97962952	486.98406982	0.13721749	0.01923917
187.73838806	119.36048889	424.72036743	0.14980558	0.01737038
160.96154785	151.51240540	431.54138184	0.15333082	0.01733854
187.54216003	119.57574463	424.74411011	0.14982831	0.01737001
310.91882324	22.92421722	460.97402954	0.13925257	0.01848049
136.00886536	187.52807617	446.29364014	0.15773122	0.01733217
34.95783615	451.98077393	901.45635986	0.20957275	0.01767208
328.35339355	13.55568695	473.15502930	0.13812403	0.01883213
320.47241211	17.69870758	467.43109131	0.13861422	0.01866591
339.87490845	7.75438690	482.19122314	0.13748769	0.01909738
187.56809998	119.54727173	424.74090576	0.14982559	0.01736999
195.08088684	111.49882507	424.07281494	0.14898418	0.01738664
163.68769836	147.96150208	430.48059082	0.15292513	0.01734017
320.47241211	17.69870758	467.43109131	0.13861422	0.01866591
91.58808136	272.66458130	510.52459717	0.17055161	0.01733634
328.97195435	13.23666382	473.61987305	0.13808717	0.01884573
143.94515991	175.33462524	440.49902344	0.15617952	0.01733289
145.76510620	172.64111328	439.32855225	0.15584560	0.01733313
320.26513672	17.80967712	467.28546143	0.13862748	0.01866170
194.60092163	112.00151062	424.10116577	0.14903617	0.01738547
209.41654968	97.15956116	424.06842041	0.14750753	0.01743094
291.91275024	34.04071045	449.65133667	0.14062098	0.01816181

Table 5.16: Population obtained at Generation 10 by NSGA For the Filter Problem

R_3	R_2	C	Variance - ω_c	Variance D
253.06492615	60.28607178	432.67526245	0.14365315	0.01769980
252.11167908	60.99897766	432.36395264	0.14373069	0.01769133
102.26777649	249.08877563	488.26702881	0.16663949	0.01733353
162.34013367	149.70817566	430.99325562	0.15312400	0.01733932
239.33801270	70.91676331	428.69888306	0.14479448	0.01759019
347.04946899	4.28646851	488.23309326	0.13715492	0.01927620
274.36953735	45.25253296	440.95932007	0.14195783	0.01792346
271.25918579	47.34527588	439.59362793	0.14220037	0.01788649
337.49560547	8.92847443	480.25875854	0.13760982	0.01904044
335.95407104	9.69569397	479.02536011	0.13769186	0.01900412
269.47863770	48.55833435	438.83563232	0.14233987	0.01786599
271.39999390	47.24981689	439.65426636	0.14218959	0.01788813
291.90246582	34.04699707	449.64578247	0.14062171	0.01816164
252.12193298	60.99128723	432.36724854	0.14372976	0.01769143
292.02932739	33.96939087	449.71472168	0.14061181	0.01816356
160.18569946	152.53559875	431.86056519	0.15344846	0.01733804
300.06533813	29.14911652	454.26321411	0.14002091	0.01829051
288.02844238	36.44004822	447.58209229	0.14091149	0.01810458
226.92475891	81.25445557	426.07388306	0.14587377	0.01751181
329.53509521	12.94699097	474.04498291	0.13805395	0.01885813
298.75415039	29.92297363	453.49682617	0.14011604	0.01826903
236.85957336	72.92333984	428.09924316	0.14500621	0.01757301
293.14074707	33.29154968	450.32266235	0.14052913	0.01818047
237.03141785	72.78329468	428.13961792	0.14499126	0.01757421
337.14700317	9.10151672	479.97857666	0.13762832	0.01903217
187.59408569	119.51875305	424.73773193	0.14982221	0.01737005
334.86026001	10.24324036	478.15908813	0.13775127	0.01897865
257.59939575	56.94422913	434.22665405	0.14328530	0.01774174
189.53137207	117.40632629	424.51907349	0.14960031	0.01737397
293.14736938	33.28753662	450.32632446	0.14052862	0.01818057

187.59675598	119.51586914	424.73739624	0.14982219	0.01737007
146.91600037	170.95675659	438.61676025	0.15563837	0.01733338
271.25885010	47.34550476	439.59344482	0.14220057	0.01788644
255.46995544	58.50355530	433.48364258	0.14345787	0.01772164
321.24374390	17.28668213	467.97525024	0.13856474	0.01868170
189.54255676	117.39419556	424.51785278	0.14959906	0.01737400
299.57583618	29.43745422	453.97595215	0.14005625	0.01828244
313.32382202	21.58704376	462.55053711	0.13908805	0.01852557
101.52647400	250.65176392	489.62725830	0.16689017	0.01733368
334.87344360	10.23661804	478.16940308	0.13775040	0.01897899
258.51773071	56.27719116	434.55496216	0.14321151	0.01775056
323.17864990	16.25942993	469.35568237	0.13844143	0.01872166
271.41064453	47.24261475	439.65887451	0.14218861	0.01788821
270.12548828	48.11636353	439.10900879	0.14228937	0.01787339
171.20315552	138.51664734	428.01919556	0.15186800	0.01734638
271.40264893	47.24801636	439.65542603	0.14218907	0.01788811
132.14320374	193.74530029	449.56036377	0.15854776	0.01733212
300.01266479	29.18011475	454.23223877	0.14002474	0.01828968
220.47259521	86.92549133	425.09436035	0.14645834	0.01747826
236.04319763	73.59037781	427.90988159	0.14507593	0.01756751
189.91592407	116.99014282	424.47958374	0.14955661	0.01737478
100.34774780	253.15878296	491.84197998	0.16729471	0.01733385
144.06031799	175.16314697	440.42330933	0.15615843	0.01733288
288.13662720	36.37260437	447.63858032	0.14090346	0.01810612
336.55868530	9.39416504	479.50738525	0.13765956	0.01901828
189.95108032	116.95214844	424.47604370	0.14955303	0.01737487
184.82955933	122.57978821	425.10839844	0.15014558	0.01736496
336.83773804	9.25525665	479.73062134	0.13764472	0.01902487
143.99142456	175.26571655	440.46859741	0.15617087	0.01733291
189.40942383	117.53849792	424.53176880	0.14961445	0.01737370
161.66555786	150.58882141	431.25845337	0.15322486	0.01733891
255.79118347	58.26719666	433.59414673	0.14343154	0.01772466
235.74098206	73.83813477	427.84072876	0.14510192	0.01756552
277.93914795	42.89108276	442.59194946	0.14168169	0.01796782
314.90975952	20.71337891	463.60797119	0.13898073	0.01855588
161.03804016	151.41186523	431.51028442	0.15331917	0.01733850
92.38404846	270.82565308	508.64950562	0.17023580	0.01733601

296.12689209	31.48825073	451.98968506	0.14030859	0.01822687
346.39788818	4.59702301	487.67089844	0.13718267	0.01925953
295.95767212	31.58976746	451.89385986	0.14032124	0.01822418
291.91256714	34.04081726	449.65124512	0.14062080	0.01816184
305.41027832	26.04389191	457.48583984	0.13963777	0.01838139
272.05590820	46.80604553	439.93841553	0.14213803	0.01789574
276.71231079	43.69789124	442.02304077	0.14177610	0.01795231
237.04069519	72.77574158	428.14184570	0.14499071	0.01757420
187.59675598	119.51586914	424.73739624	0.14982219	0.01737007
270.12524414	48.11653137	439.10888672	0.14228916	0.01787333
280.26568604	41.37466431	443.69342041	0.14150202	0.01799781
282.09255981	40.19622803	444.57901001	0.14136197	0.01802202
104.45127106	244.54483032	484.40158081	0.16591811	0.01733324
274.71035767	45.02520752	441.11218262	0.14193109	0.01792754
299.56997681	29.44091034	453.97253418	0.14005704	0.01828235
255.79118347	58.26719666	433.59414673	0.14343154	0.01772466
183.80899048	123.72384644	425.26296997	0.15026723	0.01736315
102.26777649	249.08877563	488.26702881	0.16663949	0.01733353
266.98391724	50.27673340	437.80291748	0.14253664	0.01783809
257.59939575	56.94422913	434.22665405	0.14328530	0.01774174
194.77742004	111.81646729	424.09051514	0.14901702	0.01738588
321.33349609	17.23883820	468.03881836	0.13855879	0.01868351
337.13635254	9.10681152	479.96997070	0.13762899	0.01903190
236.04319763	73.59037781	427.90988159	0.14507593	0.01756751
236.04319763	73.59037781	427.90988159	0.14507593	0.01756751
206.30323792	100.16654968	423.93603516	0.14781573	0.01741973
235.97114563	73.64942932	427.89334106	0.14508198	0.01756703
171.26773071	138.43763733	428.00082397	0.15185936	0.01734643
248.22230530	63.94593811	431.14752197	0.14405051	0.01765820
142.89353943	176.90805054	441.20120239	0.15637623	0.01733274
337.53836060	8.90726471	480.29312134	0.13760765	0.01904144
152.08801270	163.56323242	435.67745972	0.15474308	0.01733468
223.25904846	84.45025635	425.48400879	0.14620365	0.01749217

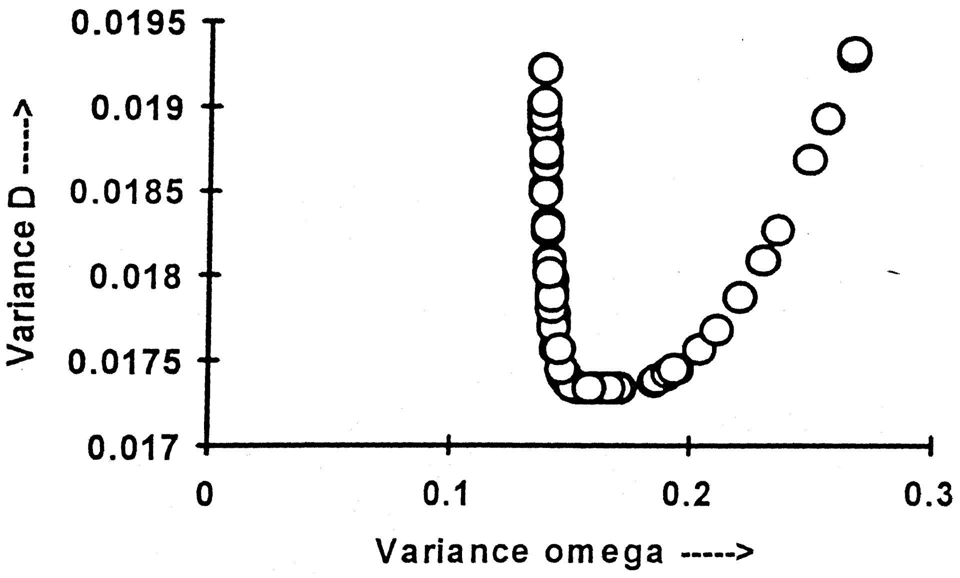

Figure 5.5: Initial Population Spread by ENGA (Filter Problem)

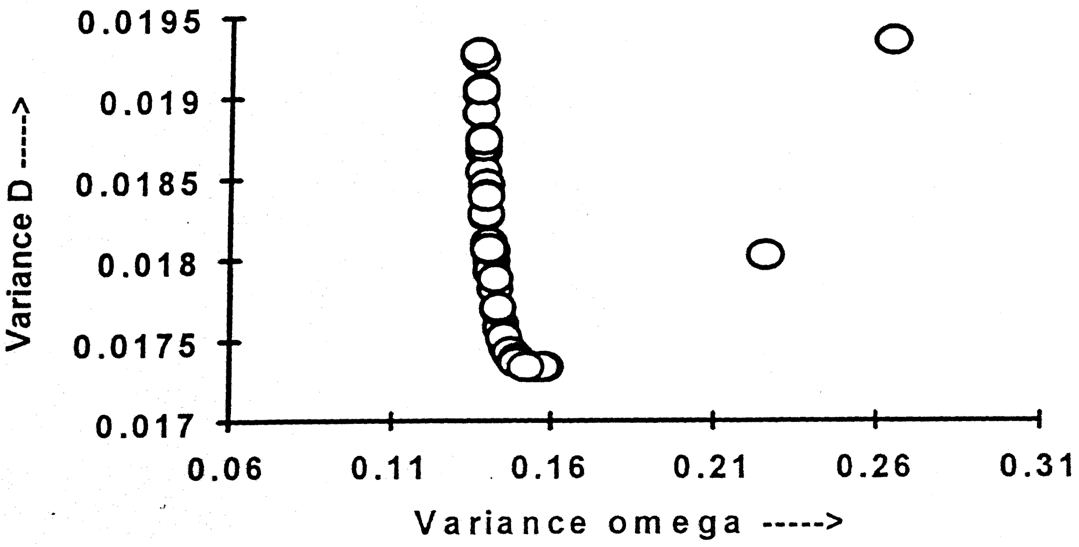

Figure 5.6: Population at Generation 5 by ENGA (Filter Problem)

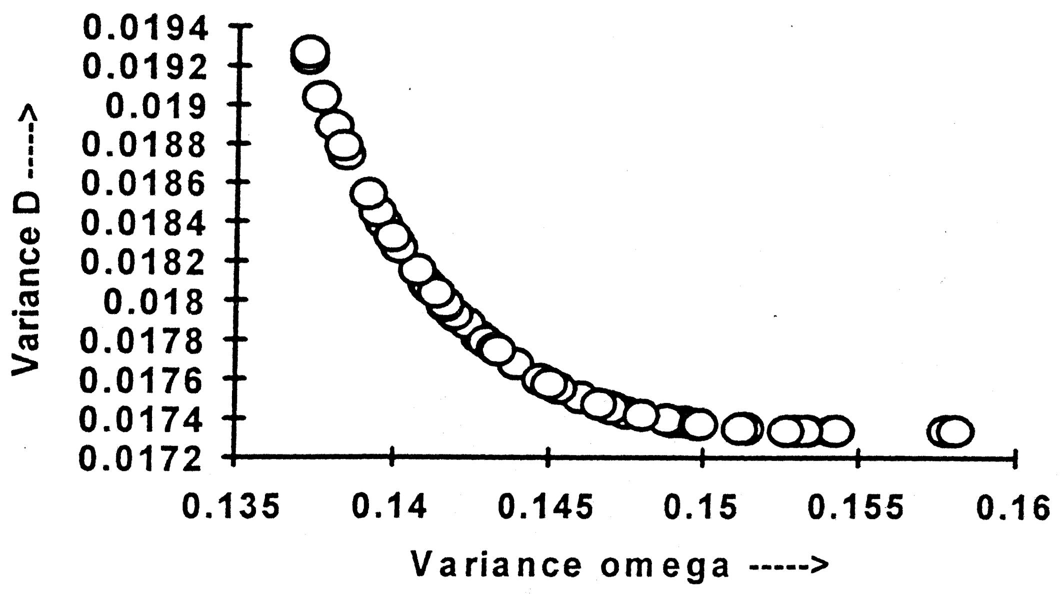

Figure 5.7: Pareto Front at Generation 10 by ENGA (Filter Problem)

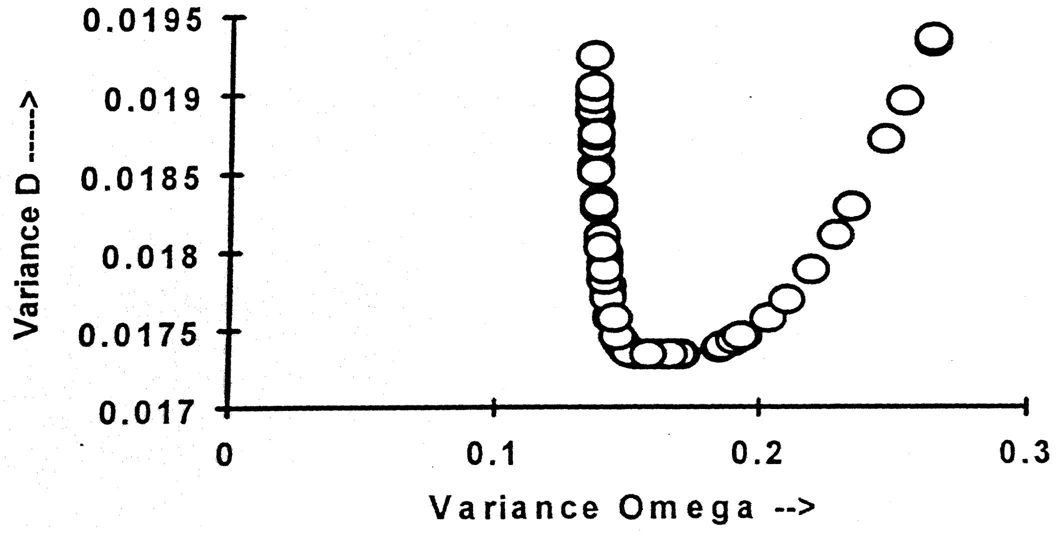

Figure 5.8: Initial Population Spread by NSGA (Filter Problem)

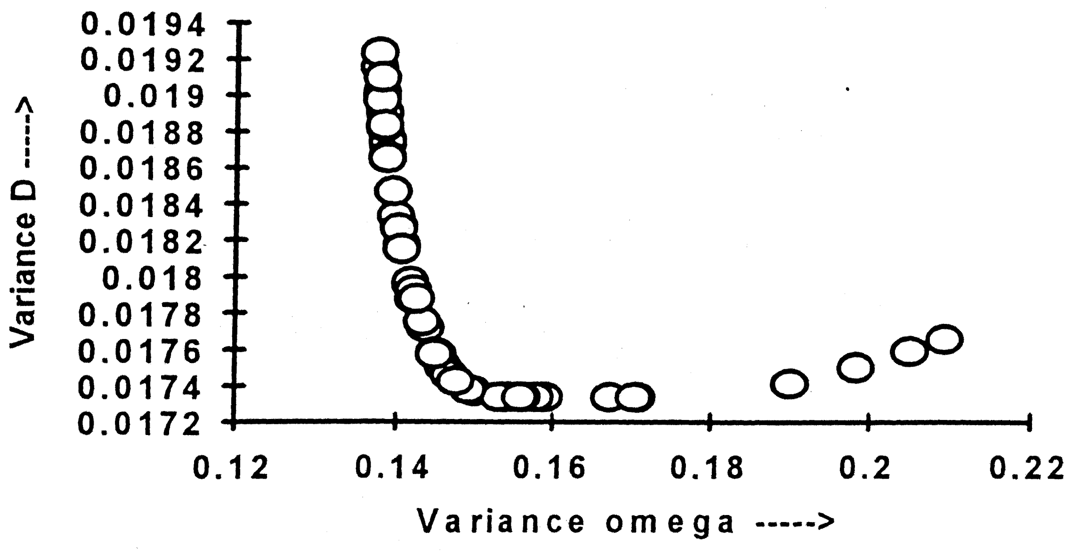

Figure 5.9: Population at Generation 5 by NSGA (Filter Problem)

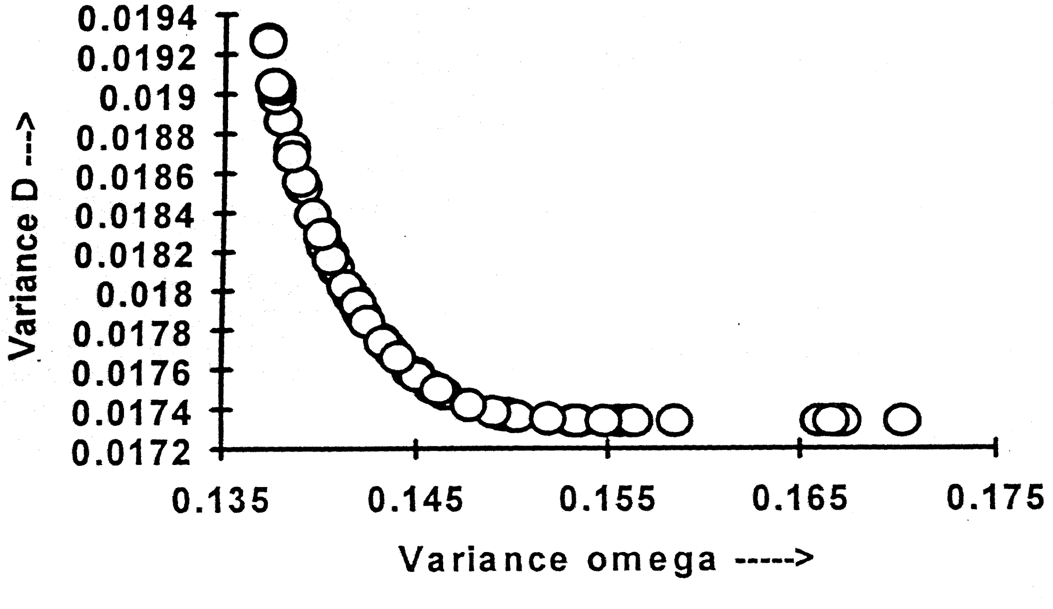

Figure 5.10: Pareto Front at Generation 10 by NSGA (Filter Problem)

Chapter 6

IMPLEMENTATION OF ENGA USING C

This chapter describes the C language implementation of ENGA (the multiobjective Elitist Nondominated Genetic Algorithm) presented in Chapter 4. The important aspects of ENGA including selection, crossover, mutation, fitness sharing and the special feature of ENGA (the nondominated sorting of the combined parents + progeny pool) are discussed. The following sections describe a bi-objective implementation of ENGA.

6.1 Input (Control) Parameter Specification

The input data to be provided to ENGA are the maximum number of generations (*maxgen*), crossover probability (*pcross*), mutation probability (*pmute*), length of the chromosome for each variable (*lsubstr*[*i*]) and the maximum and minimum bounds for each variable (*alow*[*i*] and *ahigh*[*i*]). The GA parameters — population size (*ipopsize*), number of variables (*nvar*) and overall length of the chromosome (*lchrom*)—are defined at the beginning of the program code (i.e., in the # define statements) as constant values. The subroutines *initdata*( ) and *initialize*( ) in the program read the input parameter values.

6.2 Initialization of ENGA

Initialization is done by *rgenerate*( ) subroutine of the program using the *srand*( ) library routine in C language. The user has to enter a single random integer number between 1 and 32767 and accepted as a parameter by the *srand*( ) routine. This, along with the values for all of the input parameters (Section 6.1) provided, initializes the ENGA.

6.3 Generation of Initial Population (Box -1, Figure 4.1)

The *initpop*( ) subroutine creates random strings each of specified length. This length controls the precision in the value of the variable (the larger the length more precision and vice versa). The number of initial solutions equals population size. Each digit of a solution string (0 or 1) is decided by generating a random number and by then comparing this number with the probability of each digit of the string to be a 0 or 1. The value of the probability is taken as 0.5. If the random number generated is less than 0.5 then that digit is 1 else it is 0. The *iflip*( ) routine does the generation of random number and comparison with probability and returns digit 0 or 1 accordingly. Then this population is evaluated to find the two fitness (*fitness1* [*j*] and *fitness2* [*j*]) values for each chromosome. The code for this is shown below

```
initpop ()
{
for (j = 0; j < ipopsize; j ++)
  {
     for (j1 = 0; j1 < lchrom; j1 ++)
```

```
        { chr [j1] = iflip (0.5); chrom [j][j1] = chr [j1]; }
      decodevars ( );
      for (j1 = 0; j1 < nvar ; j1 ++ )
          {oldx [j] [j1] = x [j1]; }
      fitness1 [j] = funct1 ( j);
      fitness2 [j] = funct2 ( j);
  }
return };
```

6.4 Calculation of Fitness

Recall from Chapter 2 that for maximization problems the fitness is the value of the objective function itself and for minimization some transformation is used to calculate the fitness. The generally used transformation is

fitness (F) = 1.0 / (1.0 + f (x)) where f(x) is the value of the objective function.

This change has to be made in the program depending on the optimization being sought. The *funct1*( ) and *funct2*( ) subroutines calculates the values of the two objective functions and returns either that calculated value or the transformed value as the fitness to the calling routine.

6.5 Reproduction (selection) (Box - 5, Figure 4.1)

In this program remainder roulette wheel selection method is used to form the mating pool [11]. First the expected number of copies (*expected*) for each chromosome is calculated by dividing its fitness with the average fitness of the population. Then the mantissa number of copies (*jassign*) of each chromosome are copied in the mating pool. Then the fractional part of each chromosome is considered to assign the rest of the copies until the number equals the population size. For this the fractional part (*fraction* [*j*]) is used as the probability for *iflip*( ) routine discussed in Section 6.3. If random number generated is less than the fractional part then one copy of that member is copied in the mating pool. The fractional part of this chromosome is reduced by number 1. The member is considered for selection only if its fractional part is positive. The *preselect*( ) routine does this selection

6.6 Crossover and Mutation (Box -6 and Box-7, Figure 4.1)

The program uses single point crossover. Each time two strings (*mate1* and *mate2*) are selected randomly from the population (without replacement) for crossover. The selection of a particular string is done by *iselect2*( ) routine. This uses the routine *irnd*( ) which randomly creates a integer between specified higher and lower value. For instance, for first one the low and high values are 1 and 100 when population size is 100. After each selection that "mate" is not considered for next selection.

After selection of the strings, the decision is made whether they should participate in crossover or not. For this *iflip*( ) routine is used to generate random number and compare with the crossover probability (*pcross*). If the random number is less than *pcross* then they only participate in crossover. After determining that the members participate in crossover, a random cross site (*jcross*) is determined again using *irnd*( ) routine with low and high values as 1 and the length of the chromosome (*lchrom*) respectively. The strings after crossover are the strings with the bits left to that of the crossing site (*jcross*) being the same and those on the right side interchanged.

After crossover, each time while copying a bit into the progeny, it is tested for mutation. The *mutation*( ) routine decides whether a bit has to be mutated or not. For this the *iflip*( ) generates a random number between 0 and 1 and compares it to the mutation probability (*pmute*). The bit is mutated only when the generated number is less than *pmute*. The code for crossover is shown below

```
crossover (ipop, mate1, mate2)
{
  if (iflip (pcross) = = 1)
       { jcross = irnd (1, lchrom);
       ncross = ncross + 1; }
  else jcross = lchrom;
  for (j = 0; j < jcross; j++)
       { newchr [ipop][j] = mutation ( chrom [mate1][j] );
       newchr [ipop + 1][j] = mutation ( chrom [mate2][j] ); }
  if (jcross != lchrom)
      { for ( j= jcross; j < lchrom; j++)
          {newchr [ipop][j] = mutation ( chrom [mate2][j] );
            newchr [ipop + 1][j] = mutation ( chrom [mate1][j] ); }
      }
return;
}
```

6.7 Fitness Sharing (Box - 4, Figure 4.1)

The program uses phenotypic sharing [14]. Sharing in each front is achieved by calculating a sharing function value as described in Section 3.4 of Chapter 3, between two individuals in the same front.

The phenotypic distance (*distance_1*) between two individuals in the current front is calculated by the *distance* ( ) routine. The parameter *dshare* in the program represents the maximum phenotypic distance allowed between any two individuals to become members of a niche. The parameter niche count (*anichent*) for an individual is calculated by adding the sharing function values for all individuals in the current front. Finally, the shared fitness value of each individual is calculated by dividing its dummy fitness value by its niche count. The following shows the code for sharing.

```
phenoshare ( )
double epsilon = 0.000001;   (an arbitrarily chosen small number)
{ for (j = 0; j < ipopsize; j++)
{ anichent = 1.0;
    dshare = 0.2;      /* this parameter value is for a particular problem. User has to input this
                                   value depending on the problem  */
    if (opflag [j]= = 2)
    for (i = 0; i < ipopsize; i++)
        { if (i != j)
            { if (opflag [i]= = 2) distance (j, i);
            if (distance _1 < epsilon) anichent = anichent + 1;
            else   if (distance_1 < dshare )
                 anichent = anichent + pow ((1.0-(distance_1/dshare) ), 2.0); }

      }
      fitness [j] = fitness [j] / anichent;}
return; }
```

6.8 The Additional Nondominated Sorting in ENGA (Box - 8 and Box - 9, Figure 4.1)

This procedure consists of mixing and nondominated sorting of the *combined* population of new and old individuals. Subsequently, the top 50 % of the population after sorting is taken as the progeny population. The fitness values of the combined population is stored in the arrays *engafitness1* [ ] and *engafitness2* [ ]. The following code shows the procedure for selecting top 50 % after nondominated sorting.

```
j = 0; k = 1;
while (j < ipopsize)
 {
    for (i = 0; i < (2 * ipopsize); i++)
       { if ((opfront [i]= = k) && (j < ipopsize))
          {
             fitness 1[j] = fitness 1 [i];
               engafitness 1[j] = engafitness 1 [i];
```

```
            fitness 2[j] = fitness 2 [i];

            engafitness 1[j] = engafitness 1 [i];

                for (l = 0; l < lchrom; l++)

                {

                    newchr [j][l] = engachrom [i][l];

                    engachrom [j][l] = engachrom [i][l];

                }

            j = j+1;

        }

    }

    k = k + 1;

}
```

6.9 Termination Criteria and Output

The program produces the following output.

- The program prints the following values for all the members of the population for each generation.
  - The values of the variables.
  - The values of the objective functions.
  - The front numbers.
- Maximum, average and minimum values of the fitness of the population.
- The best chromosome, its fitness and the generation at which it was found.
- The number of crossovers and mutations.

The program terminates when the maximum generation number specified (*maxgen*) is reached. In this chapter only the relevant logical parts of the code have been described. These, when combined with variable declarations and integrating these in the appropriate manner with each other and with the main body , produces the complete executable program (see Appendix). The other routines i.e., those decoding variables and those to calculate and print population statistics, and the utility subroutines are self-explanatory and hence these are not discussed in this chapter.

Chapter 7

CONCLUSIONS AND DIRECTIONS FOR FUTURE WORK

In Chapter 1 we reviewed the multi-objective optimization problem formulated as the multiple criteria decision making (MCDM) problem. A key feature of the MCDM is the conflicting nature of the objectives to be optimized. Traditional methods to solve the multi-objective optimization problems were also reviewed, along with their limitations. The concept of Pareto optimality and efficient solutions was introduced.

Chapter 2 provided a brief introduction to Genetic Algorithms, a set of meta-heuristic methods that are fast emerging as efficient search and optimization algorithms. The features and the structure of the GA were summarized. The merit of GAs over traditional optimization methods and some advanced variations of GAs were also indicated.

Chapter 3 discussed the Nondominated Sorting Genetic Algorithm (NSGA). The logic of NSGA and the concepts of fitness sharing and nondominance were explained. A bi-objective robust electronic filter design problem was reworked to illustrate the application of NSGA.

Chapter 4 described ENGA, the enhanced multi-objective GA developed and evaluated in this work. To illustrate its use ENGA was applied to solve a two-objective engineering optimization problem. The statistical design of experiments (DOE) framework was used to find the individual factor effects and the interactions among the affects of the various ENGA parameters. Full factorial experiments were conducted to find optimum ENGA parameter settings.

In Chapter 5 the performance of NSGA was compared statistically with that of ENGA. This comparison was done on the basis of two distinct performance criteria: (a) Number of function evaluations to obtain 95% of the population on the nondominated front (Front 1) and (b) Number of solutions on the Front 1 when criteria 1 has been satisfied. NSGA and ENGA were compared for each of these two criteria using the Wilcoxon signed-rank test. ENGA proved to be more efficient.

In Chapter 6 the C code implementation of ENGA accomplished in this work was described. The genetic operations —reproduction, crossover, mutation and fitness sharing—and the implementation of the additional nondominated sorting feature of ENGA were explained.

The study has led to following conclusions:

(i) Most Traditional methods appear to solve multi-objective optimization problems by converting the multiple objectives into single objective by suitable means. Pareto optimality may be obtained for some well behaved problems using the ε-method available in the literature. However, no general method takes care of optimizing all the objectives simultaneously to yield, for instance, a set of Pareto optimal solutions.

(ii) The nondominated sorting technique appears to apply effectively to multi-objective optimization problems. It delivers Pareto optimal solutions, subsequent to which the decision maker may choose a particular solution depending on his/her requirements and preferences. Meta-heuristic methods such as NSGA can do the requisite here.

(iii) Genetic Algorithms (SGA, NSGA and ENGA) yield their best performance for certain combinations of parameters only. The optimum selection of parameters definitely increases the efficiency of GAs. Therefore, a systematic multifactor search procedure such as *design of experiments* should be applied for finding appropriate optimum values for the GA parameters. one-factor-at-a-time methods are ineffective here.

(iv) The ENGA algorithm described in this work for multiobjective optimization appears to take significantly fewer function evaluations than NSGA (a method made recently available in the literature) to discover the nondominated member. This is supported by statistical tests done on typical problem data. Thus the convergence of ENGA to efficient solutions is indicated to be faster than NSGA.

(v) By the time NSGA is able to put say 80% of the members on the nondominated (efficient) front, ENGA obtains almost all the members on that front. This feature holds valid for two bi-objective problems tested. ENGA solutions appear comparable in quality, but are superior in number, to NSGA solutions. On the other hand, NSGA appears to retain many duplicate solutions resulting in clustering of solutions on the Pareto front.

(vi) Multi-objective solutions obtained by nondominated sorting methods (NSGA and ENGA evaluated here) can indeed free the decision maker from the drawbacks of traditional weighted-sum one-objective approaches which require decision maker's preferences to be stated *a priori*.

Suggestions for Future Work

- This study used the Roulette wheel selection and single point crossover for both NSGA and ENGA. Based on literature, one would expect an improvement in computational effort and in the quality of the solutions using operators such as tournament selection, uniform crossover etc. This may be further explored.

- The effectiveness of ENGA over NSGA may be tested with problems involving more than two objectives, and also for a wider class of multi-objective optimization problems, beyond the two problems addressed here.

- In the present work, in ENGA, after nondominated sorting of the combined (parents+progeny) pool was done, the top 50% of the combined pool was taken as the population for the subsequent generation. The effects of varying this "elite fraction" of the combined pool to be taken as the population for the next generation may be studied.

- The present work has used only binary coding of decision variables in the application of ENGA. The application of ENGA should be further explored using real coded decision variables—to evaluate ENGA's efficiency over NSGA in finding Pareto optimal solutions.

- The present study indicated two-level interactions existing among the GA parameters. There are indications that higher level interactions among these parameters also may be significant. These may be investigated to a finer degree using higher resolution experimental designs.

- An ENGA-based commercial software may be developed to assist in the solution of multiobjective problems.

REFERENCES

[1] Adulbhan, P. and M. T. Tabucanon (1980). Multicriterion Optimization in Industrial Systems. Chapter 9, Decision Models for Industrial Systems Engineers and Managers, AIT, Bangkok, Thailand.

[2] Bagchi, Tapan P. (1993). *Taguchi methods Explained - Practical Steps to Robust Design*, Prentice Hall of India.

[3] Bagchi, T. P. and Kumar (1993). Multi-criteria Robust Design of Electronic Devices, *Journal of Electronics Manufacturing*, vol 3, 31-38.

[4] Bagchi, T. P. (1997). Lecture Notes on Genetic Algorithms

[5] Bagchi, T. P. (1997) . Personal Communication

[6] Bendell, T. (1989). Taguchi methods, in *Proceedings of the 1988 European Conference*, Elsevier applied Science, New York.

[7] Box, G. E. P. (1957) . Evolutionary Operation: A Method for Increasing Industrial Productivity, *Applied Statistics*, Vol. 6, 81-101.

[8] Box, G. E. P. (1988). Signal-to-Noise Ratios, Performance Criteria, and Transformations, *Technometrics*, 30(1), 1-40

[9] Cochrane, J. L. and M. Zeleny, eds., (1973). *Multiple Criteria Decision Making*, University of South Carolina Press.

[10] Connolly, D. T. (1992). General Purpose Simulated Annealing, *J. ORSA*, 43, 3, 495-505.

[11] Deb, K. (1995). *Optimization for Engineering Design - Algorithms and Concepts*, Prentice Hall of India.

[12] Deb, K. and Srinivas, N. (1995). Multiobjective Optimization using Nondominated Sorting Genetic Algorithm, Massachusetts Institute of Technology. *Evolutionary Computations*, 2(3), 221-248

[13] Deb, K. (1997). Personal Communication

[14] Deb, K. and Goldberg, D. E. (1989). An Investigation of Niche and Species Formation in Genetic Function Optimization, *International Conference on GA*, 97-106.

[15] Dehnad, K. (1989). *Quality Control, Robust Design, and the Taguchi Method* , Wadsworth, CA.

[16] De Jong, K. A. (1993). Genetic Algorithms are NOT Function Optimizers, *FOGA*, 5-17.

[17] De Jong, K. A. (1975). An Analysis of the Behaviour of a Class of Genetic Adaptive Systems, *Doctoral Thesis*, Department of Computer and Communications Sciences, University of Michigan, Ann Arbor.

[18] Dowsland, K. A. (1996). Genetic Algorithms - a Tool for OR, *Journal of the Operational Research society*, 47, 550-561.

[19] Flippone, S. F. (1989). Using Taguchi Methods to Apply to Axioms of Design, *Robot, Computer Integrated Manufacturing,* 6(2), 133-142

[20] Ghosh, S. (1990). *Statistical Design and Analysis of Industrial Experiments,* Marcel Dekker Increase., New York.

[21] Glover, F. (1990). Tabu Search- Part I, *ORSA Journal on Computing 1*, 3, 190-206.

[22] Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York.

[23] Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-16, 122-128.

[24] Holland, J. (1975). *Adaption in Natural and Artificial Systems,* University of Michigan Press.

[25] Holland, J (1992). Genetic Algorithms, *Scientific American,* 15-21.

[26] Horn, J. N. Nafpliotis, and Goldberg, D. E. (1994). A Niched Pareto Genetic Algorithm for Multiobjective Optimization, *IEEE,* 82-87.

[27] Hwang, C. L. , A. S. M. Masud, S. R. Paidy, and K. Yoon (1982). Mathematical Programming with Multiple Objectives: A Tutorial. Computers and Operations Research: *A Special Issue on Mathematical Programming with Multiple Objective,* Vol. 7, No. 1-2.

[28] Hwang, C. L. and A. S. M. Masud (1979). Multiple Objective Decision Making-Methods and Applications: A state-of-the-art Survey. Springer-Verlag.

[29] Johnson, E. (1968). Studies in Multi-objective Decision Models, Monograph No. 1, Economic Research Centre, Lund, Sweden.

[30] Kackar, R. N. (1985). Off-line Quality Control, Parameter Design and the Taguchi Method, *Journal of Quality Technology,* 17(4), 176-209.

[31] Keeney, R. (1983). *Decisions with Multiple Objectives-Preferences and Value Trade-offs,* John Wiley and Sons.

[32] Koza, John R. (1994). *Genetic Programming II,* MIT Press, Cambridge, MA.

[33] Kumar, S. (1997). An Application of Genetic Algorithms in No-Wait Lot Streaming Flowshop Scheduling, *M.Tech. Dissertation,* IME, IIT Kanpur.

[34] Lietmann, G. and A. Marzollo, eds., (1975). *Multicriteria Decision Making,* Springer-Verlag.

[35] Montgomery, D. C., George, C. R. (1994). *Applied Statistics and Probability for Engineers,* John Wiley and Sons Increase, New York.

[36] Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines, *Journal of Chemical Physics*, 21, 1087-1092.

[37] Montgomery, Douglas C. (1997). *Design and Analysis of Experiments*, John Wiley.

[38] Mitchell, M. (1996). *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA.

[39] Phadke, M. S. (1989). *Quality Engineering and Robust Design,* Prentice Hall , New Jersy.

[40] Prasad, J. (1997). A Study in Multi-Response Robust Design Using Genetic Algorithms, *M.Tech. Dissertation*, IME, IIT Kanpur.

[41] Raman, K. (1994). An Evaluation of Genetic Algorithms for Robust Design, *M.Tech. Dissertation*, IME, IIT Kanpur.

[42] Roy, B. (1971). Problems and Methods with Multiple Objective Functions, *Mathematical Programming*, 1, 239-266.

[43] Seo, Fumiko and Masatoshi Sakawa (1988). *Multiple Criteria Decision Analysis in Regional Planning : Concepts, Methods and Applications*, Reidel, Boston.

[44] Sharma, P. (1996). Multicriteria Robust Design Using Nondominated Sorting Genetic Algorithms, *M. Tech. Dissertation*, IME, IIT Kanpur.

[45] Simon, H. A. (1969). *Science of the Artificial*, MIT Press.

[46] Spillman, R. (1993). Genetic Algorithms- Nature's Way to Search for the Best, *Dr. Dobb's Journal*, 26-30.

[47] Srinivas, M. and L. M. Patnaik (1994). Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, 4, 656-667.

[48] Suh, N. P. (1990). *The Principles of Design,* Oxford University Press, New York.

[49] Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms, *ICGA*, 2-9.

[50] Syswerda, G. (1993). Simulated Crossover in Genetic Algorithms, *FOGA*, 239-255.

[51] Tabucanon, M. T. (1989). *Multiple Criteria Decision Making in Industry*, Elsevier Science Publishers, New York.

[52] Taguchi, G. (1986). *Introduction to Quality Engineering*, Asian Productivity Organization, Tokyo.